A SERVICE ORIENTATED ARCHITECTURE AND WIRELESS SENSOR NETWORK APPROACH APPLIED TO THE MEASUREMENT AND VISUALISATION OF A MICRO INJECTION MOULDING PROCESS

UMAR RAZA

PhD

2014

A SERVICE ORIENTATED ARCHITECTURE AND WIRELESS SENSOR NETWORK APPROACH APPLIED TO THE MEASUREMENT AND VISUALISATION OF A MICRO INJECTION MOULDING PROCESS

Design, Development and testing of an ESB based Micro Injection Moulding platform using Google Gadgets and Business processes for the integration of disparate hardware systems on the factory shop floor

Umar Raza
BSc(Hons),MSc

Submitted for the degree of
Doctor of Philosophy

School of Engineering and Informatics

University of Bradford

2014

Abstract

Factory shop floors of the future will see a significant increase in interconnected devices for monitoring and control. However, if a Service Orientated Architecture (SOA) is implemented on all such devices then this will result in a large number of permutations of services and composite services. These services combined with other business level components can pose a huge challenge to manage as it is often difficult to keep an overview of all the devices, equipment and services.

This thesis proposes an SOA based novel assimilation architecture for integrating disparate industrial hardware based processes and business processes of an enterprise in particular the plastics machinery environment. The key benefits of the proposed architecture are the reduction of complexity when integrating disparate hardware platforms; managing the associated services as well as allowing the Micro Injection Moulding (μIM) process to be monitored on the web through service and data integration. An Enterprise Service Bus (ESB) based middleware layer integrates the Wireless Sensor Network (WSN) based environmental and simulated machine process systems with frontend Google Gadgets (GGs) based web visualisation applications. A business process framework is proposed to manage and orchestrate the resulting services from the architecture.

Results from the analysis of the WSN kits in terms of their usability and reliability showed that the Jennic WSN was easy to setup and had a reliable communication link in the polymer industrial environment with the PER being below 0.5%. The prototype Jennic WSN based μIM process monitoring system had limitations when monitoring high-resolution machine data, therefore a novel hybrid integration architecture was proposed. The assimilation architecture was implemented on a distributed server based test bed. Results from test scenarios showed that the architecture was highly scalable and could potentially allow a large number of disparate sensor based hardware systems and services to be hosted, managed, visualised and linked to form a cohesive business process.

Keywords: Micro Injection Moulding; WSN; SOA; Business Processes; BPEL, Process Monitoring; Web services, ESB, Google Gadgets

Dedicated to my beloved grandfather, ***Haji Fazal Karim*** who was the major source of inspiration and who instilled an interest in learning and an appetite for knowledge in all our family, I miss him a lot…

Acknowledgments

First and foremost, I would like to thank Almighty Allah (God) the most gracious, the most merciful, and the most compassionate for providing me the blessings, courage, strength and help to complete this work.

I would like to take this opportunity to express my sincere gratitude, gratefulness and appreciation to my principle supervisor, Dr. Ben Whiteside for taking me into the MNT research group and providing me with a great amount of freedom to work. His confidence and trust in my abilities were the major motivating factors for this achievement. I would like to thank him for his valuable advice, patience, time and effort, expert guidance and useful feedback throughout my PhD journey. In fact, without his encouragement and great support, I would not have been able to achieve this work. I would also like to thank my second supervisor Prof. Fun Hu for her exceptional support, guidance and encouragement throughout this research and for aiding me to complete this work. Additionally I would also like to take this opportunity to thank Prof. Phil Coates for his continuous encouragement and support as well as offering my appreciation to all staff, technicians and friends at the IRC University of Bradford.

The major contributors to this achievement are my parents and I would like to express my deep gratitude for their continuous support, advice, patience and motivation. It has been their prayers, encouragement and support that enabled me to complete this research.

Also I am indebted and grateful to my brother Dr. Mansoor Raza for helping me throughout this long journey. His constant support, useful discussions, and review of my research provided me with encouragement and an integral view of the research.

Finally I owe my deepest gratitude to my wife and lovely children who have been a great source of motivation and inspiration and whose unceasing support and patience helped ensure the success of this thesis. I really appreciate their patience for accepting my absence for long hours which must have spoiled some precious moments of our life

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

List of Acronyms

6LoWPAN	IPV6 over Low-Power WPAN
µIM	Micro-Injection Moulding
µIMP	Micro-Injection Moulding Process
ADC	Analogue to Digital Convertor
API	Application Programming Interface
APL	Application layer
AS	Application Server
ASCII	American Standard Code for Information Interchange
B2B	Business to Business
BC	Binding Component
BPEL	Business Process Execution Language
BPM	Business Process Model
BPS	Business Process Server
BT	Bluetooth
CAN	Controller Area Network
CDAQ	Compact DAQ
CES	Complex Event Server
CGI	Common Gateway Interface
CMOS	Complementary Metal Oxide Semiconductor
CORBA	Common Object Request Broker Architecture
COM	Component Object Model
COTS	Commercial, off-the-shelf
CPU	Central Processing Unit
CRIO	Compact Reconfigurable Input Output
CRM	Customer Relationship Management
CSV	Comma Separated Values
DAC	Digital to Analogue Convertor
DAQ	Data Acquisition
DB	Database
DOM	Document Object Model
DPWS	Device Profile Web Service
DSS	Data Service Server
DTD	Document Type Definition
EDA	Event-Driven Architecture
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIS	Enterprise Information System
ER	Error Rate
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
FMEA	Failure Modes and Effects Analysis
FPGA	Field Programmable Gate Array

FRM	Financial Resource Management
FTDI	Future Technology Devices International
GGs	Google Gadgets
GS	WSO2 Gadget Server
GUI	Graphical User Interface
GWELs	Graphical Workflow Execution Language for Sensor Networks
HMI	Human-Machine Interaction
HP	High Power
HR-WPAN	High Rate WPAN
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
I2C	Inter-Integrated Circuit
ICMP	Internet Control Message Protocol
IETF	The internet Engineering Task Force
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IM	Injection Moulding
IoT	Internet of Things
IP	Internet Protocol
IP-CAN	IP-Connectivity Access Networks
IPO	Intelligent Process Optimisation
IPv6	IP version 6
ISDN	Integrated Service Digital Network
ISM	Industrial Scientific and Medical
ISO	International Standards Organization
IT	Information Technology
JDK	Java Development Kit
JMS	Java Messaging Service
JRMP	Java Remote Method Protocol
JSON	JavaScript Object Notation
JSP	JavaServer Pages
JVM	Java Virtual Machine
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LoS	Line of Sight
LP	Low Power
LQI	Link Quality Indicator
LR-WPAN	Low Rate WPAN
LVCDA	LabVIEW Compact-DAQ Application
LVCRA	LabVIEW Compact-RIO Application

LVWSM	LabVIEW Wireless Sensor Monitor
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical Systems
MOM	Message Oriented Middleware
MNM	MoltoSenso Network Manager
MNT	Micro and Nano Technology
MR	Message Relay
MRP	Manufacturing Resource Planning
NI	National Instruments
NWK	Network Layer
NHTTP	Non-blocking HTTP
ODBC	Open Database Connectivity
OPC	Open Platform Communication
OPC UA	Open Platform Communication Unified Architecture
ORPC	Object-oriented Remote Procedure Call
OS	Operating System
OSGi	Open Services Gateway initiative
OSI	Open System Interconnection
PAT	Process Analytical Technology
PC	Personal Computer
PDR	Packet Delivery Ratio
PER	Packet Error Rate
PHP	Hypertext Preprocessor
PLC	Programmable Logic Controller
POX	Plain Old XML
PRR	Packet Reception Rate
PT-HTTP	Pass-Through HTTP
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
RPC	Remote Procedure Calls
REST	Representational State Transfer
REWISE	Reconfigurable Wireless Intelligent Sensor Networks
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SCM	Supply Chain Management
SDK	Software Development Kit
SEM	Scanning Electron Microscopy
SNMP	Simple Network Management Protocol
SMA	Sub Miniature version A
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture

SOAP	Service Oriented Architecture Protocol
SOPC	System On Programmable Chip
SPI	Serial Peripheral Interface
SS7	Signalling System #7
SQL	Structured Query Language
SWE	Sensor Web Enablement
TI	Texas Instruments
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UDDI	Universal Description, Discovery and Integration
UDL	Universal Data Link
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UWB	Ultra Wide Band
VI	Virtual Instrument
VISA	Virtual Instrument Software Architecture
VLSI	Very Large Scale Integration
WAN	Wide Area Network
WCAN	Wireless Control Area Network
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Service Definition Language
WSN	Wireless Sensor Network
WVM4AC	Wireless Medium Access Control
WWAN	Wireless Wide Area Network
XML	eXtensible Markup Language
XSD	eXtensible Schema Definition
XSLT	eXtensible Stylesheet Language Transformation

Chapter 1: Introduction

The advent of the “Internet of Things (IoT)” concept which envisages millions of devices being interconnected, providing and consuming information available on the network and co-operating (Spiess et al., 2009a), has made the Service Orientated Architecture (SOA) paradigm as the most common integration approach for integrating various sensors and devices with the internet. The last few years has seen the prevalence of the IoT concept in non-critical home and office environments, but it is only until recently that this concept has expanded into the Industrial domain (Spiess et al., 2009a, Guinard and Trifa, 2009b, Gusmeroli et al., 2012). One industrial environment which still has not had much exposure is the plastics machinery environment consisting of a combination of Injection Moulding (IM), Micro-Injection Moulding (µIM) and/or extrusion lines. The µIM process in particular has not been exposed to the IoT concept using WSNs and SOA paradigm. Latest developments in µIM machine technology allow process information to be sent over wired network links, but the resolution of key process data is typically too low for adequate characterisation of micro-moulding processes due to the small time scales involved. In light of this, the IM, µIM, and/or extrusion line manufacturing processes would directly benefit from the IoT concept by making the environmental and high-resolution process data available in real-time over the internet. This would have numerous benefits in terms of process monitoring, material and product quality.

To use the IoT concept to monitor and study the plastics machinery environment, it is necessary to identify the most common key parameters (environmental and machine) in this environment used in previous research. Based on these parameters a WSN kit needs to be selected which can meet the requirements of monitoring the identified parameters. For the last decade and more, there have been numerous studies using WSNs in industrial environments. In these studies WSN kits with varying specifications from various vendors have been used. To date there has been no comprehensive

review of these kits being tested in industrial settings in particular the plastics machinery environment.

This thesis first identifies the most common parameters used in monitoring the plastics machinery environment. Based on these parameters three WSN kits are selected from a total of nine kits. The three selected WSN kits are then evaluated for their usability and reliability in industrial and home/office environments. The thesis then focuses on further evaluating the remaining two WSN kits in terms of their data communication reliability in a home/office and plastics machinery environment using the Packet Error Rate (PER), Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) parameters. Based on these results the Jennic JN5148 WSN kit (Jennic Ltd, 2010b) is selected to be used to monitor a µIM machine and the plastics machinery environment through the use of the SOA and Web Services. A network of the Jennic JN5148 wireless sensor nodes is installed in the Battenfeld Microsystem 50 µIM machine and the plastics micro-moulding environment of the Polymer MNT Laboratory at the University of Bradford.

The next part of the thesis focuses on the novel architecture, design, development, and implementation of the LABVIEW Wireless Sensor Monitor (LVWSM) µIM process monitoring system. This system takes advantage of the SOA and applies it to the Jennic wireless sensor nodes installed in the µIM machine and the industrial micro-moulding environment. Process data is transmitted wirelessly using a sensor node to a central gateway which converts and presents the data as a service through the use of Restful Web Services. In addition to the sensor node firmware modification a new hardware interface is designed to allow the Jennic wireless sensor nodes to communicate with the µIM machine sensors. The results from the LVWSM software show that WSNs are good for monitoring the low-resolution plastics machinery environment and business processes but not fast enough for high-resolution machine data. Therefore there is a need for a wired high-speed data acquisition hardware for high-resolution machine data such as Compact-RIO from National Instruments (National Instruments, 2012d). This high-resolution machine data gives rise to a new but interesting problem of how to integrate it with the low-resolution WSN environmental and business process data?

This could be achieved through the implementation of the SOA paradigm by converting both the low-resolution WSN (environmental and business process data) and high-resolution machine data into web services. Although in this thesis we are dealing with only the environmental and machine sensor data, if the SOA paradigm is applied to each and every sensor in a large factory (Pautasso et al., 2008, Guinard and Trifa, 2009b) this will result in a large number of services and composite services driving a large number of other services, business processes and even applications. In the future this will be a potential issue as the resulting plethora of information extracted through the use of services from the various sensors in the environment and machines can become difficult to manage and requires an optimal way to categorize and make it available to multiple systems. This could be achieved through a middleware layer approach which provides a mechanism for mediation to simplify the task of connecting distributed systems. With this in mind the thesis proposes the use of the Enterprise Service Bus (ESB)(Apache, 2012d) as the enabling middleware technology for implementing the SOA. A novel four layer architecture called the Micro Injection Moulding Process (μIMP) Monitor is proposed for allowing the integration of WSN environmental data with high-resolution machine data. This architecture for monitoring the μIM process uses a distributed server architecture based on SOA with the Enterprise Service Bus (ESB) at its core. The architecture allows disparate hardware systems to be integrated and presented under a unified web portal through the use of the Google Gadgets API (Google Inc., 2011) and web visualisation and charting libraries.

Each system in the μIMP Monitor architecture will have a number of services depicting the functionality of that system. In order to allow these systems to be linked and provide some meaningful information as a single unified process, a new integration method is presented through the use of Web Services Business Process Execution Language (WS-BPEL). In this method the functionality of each stage of the μIM process and other business processes of the enterprise is depicted through use sensor technology and Web services. WS-BPEL acts as the glue to bind the services into a single unified business process. The thesis finally proposes a business processes framework. The framework allows the use of WS-BPEL to manage and orchestrate the resulting

services from the various layers of the µIMP Monitor architecture. Business processes created using WS-BPEL are used to process and analyse the data on a distributed architecture as well as giving the capability to link with other standard business processes. The framework is tested using the LVWSM software and simulated business processes.

In this thesis, extensive research and development work has been carried out to realize the LVWSM software and the novel µIMP Monitor architecture, such as WSN, ESB, WS-BPEL business processes, Google Gadgets, web visualisation libraries, databases and LABVIEW software simulation etc. This chapter will give the background of this research, raise the aims and objectives, summarise the author's contributions, give the structure of the thesis and finally the publications generated so far as a result of this PhD study.

1.1 Background to the Research Problem

The past decade and more has seen wireless technology solutions based on wireless standards like Wi-Fi, Bluetooth or ZigBee being successfully used in home, office and healthcare environments. These environments are mostly composed of insulators like wood, glass, plaster, bricks, plastics etc. On the other hand in industrial environments like the factory shop floor, these wireless technologies are considered not very well suited since these environments are radio-hostile and most of the time consist of conductors like metals, steels, cast iron, copper etc. (Assous et al., 2009). Therefore up until now the sensors, actuators, and I/Os of a factory shop floor are still connected using a separate twisted shielded pair cable to relay the gathered information to the controllers in the system. The industrial automation systems still use wired communication networking technologies based on Fieldbus systems (Pleinevaux and Decotignie, 1988), (Mahalik, 2003), (Zurawski, 2004) like PROFIBUS (Jecht et al., 2004), WorldFIP (Thomesse, 2004), or CAN (Hanxing and Jun, 2009).

Wired industrial networks are very reliable but they do have their downside of being very costly and having "information black out areas" which are inaccessible by any industrial networks due to the cabling constraints (Li, 2006, Low et al., 2005). To overcome the problems of "information black out areas" and high costs, industry is turning to wireless industrial systems

composed of Wireless Sensor Networks (WSN) which are made up of distributed intelligent nodes that communicate data using electromagnetic waves as their transmission medium.

The bulk of the research on WSN in industrial environments has focused on issues such as Quality of Service (QoS) (Tahir and Javed, 2009, Koubaa et al., 2009, Xu et al., 2009), resource constraints (Chuan-Yu et al., 2008, Xu et al., 2009), real-time data processing and delivery (Willig, 2008, Gungor and Hancke, 2009, Gungor et al., 2008), data and network reliability (Hanssmann et al., 2009, Willig, 2008), network scalability and interference factors in an industrial setting (Chilo et al., 2009, Gnad et al., 2008) to name a few. The results from these varied research efforts have given rise to a myriad of WSN applications using different network protocols and developed utilizing a number of proprietary hardware and software platforms. This has resulted in the lack of interoperability between these platforms and other machinery and equipment on the factory shop floor; as well as the problem of sensory data fusion when being integrated into applications for a meaningful purpose.

In an industrial automation domain a number of processes could be running at any one time as part of an overall manufacturing process. Each of these processes could have specific sensory inputs which can be provided only by specific hardware platforms. With this in mind a strong trend has emerged in designing WSNs composed of heterogeneous sensor devices which could be used for a wide range of large and complex applications like Enterprise-IT systems.

Earlier Enterprise-IT and Internet applications had difficulty in interacting with other applications due to their proprietary nature and hence faced similar problems as the WSNs such as interoperability and data fusion. The adoption of the Service Orientated Architecture (SOA) an architectural concept which separates functions into distinct units called services (Erl, 2009) saw the elimination of these problems in Enterprise-IT systems. Since then the SOA paradigm has become the de-facto solution in the Enterprise-IT systems and the internet as it has the benefits of modularity, flexibility, loose-coupling and interoperability. With many of the latest enterprise and internet applications

being increasingly built around the SOA architecture researchers in the field of WSNs have also started to adopt the SOA paradigm to solve the above mentioned issues hence resulting in a number of research efforts and solutions (Savio and Karnouskos, 2008, Sleman and Moeller, 2008, Samaras et al., 2009, Spiess et al., 2009a, de Souza et al., 2008). Also the advent of the “Internet of Things (IoT)” concept which envisages millions of devices being interconnected, providing and consuming information available on the network and co-operating (Spiess et al., 2009a), has made the SOA paradigm as the most common integration approach for integrating various office and home based sensors and devices with the internet. The last few years has seen the prevalence of the IoT concept in non-critical home and office environments, but is only until recently that we have started to see this concept expand into the Industrial domain (Spiess et al., 2009a, Guinard and Trifa, 2009b, Gusmeroli et al., 2012). One industrial environment which still has not had much exposure is the plastics machinery environment consisting of a combination of Injection Moulding (IM), Micro-Injection Moulding (μIM) and/or extrusion lines. The μIM process in particular has not been exposed to the IoT concept using WSNs and SOA paradigm.

Micro-injection moulding (μIM) is a variant of the injection moulding processes which allows mass production of micro- and nano-featured components for a diverse range of markets including healthcare (diagnostics, implants, drug delivery), micro optics and automotive. Figure B.2 shows the different stages in the life cycle which a product has to go through when it is manufactured using IM. The challenges associated with implementing a successful μIM process can be significant when compared with standard injection moulding. Clean material handling environments, high accuracy injection technologies, precision tooling and accurate thermal control are all required to ensure success, but even then, external variables can contribute to process failure. As a result, many of these processes require on-line Quality Assurance (QA) measurement systems which typically consist of product imaging and handling systems. Such systems can be expensive to implement and only provide information about the net shape of the product, rather than material consistency, so there is an opportunity to generate further quality

information by collecting data from each stage in the process including material preparation, melting and injection.

The factory of the future will require the amalgamation of industrial processes, WSN, machinery, and equipment on the shop floor with enterprise level applications through the implementation of the SOA paradigm. The preferred standards-based way to realize the SOA paradigm has been through web services and this provides an effective means of managing distributed services. The SOA paradigm can be applied to the individual sensor nodes at the device level or at the gateway level (see section 2.6.2). Processing of low-resolution data, could be carried out at device level, as they have limited on-board resources and this is better carried out using RESTful (Fielding, 2000) Web services. However, with high-resolution data, the limited on-board capabilities become rate limiting and therefore a central gateway architecture with higher resources which could use both WS*- Web and RESTful services (Guinard and Trifa, 2009a, Pautasso et al., 2008) could be the way forward. In either case if every sensor in the factory environment, machines and equipment is abstracted as a service then this will result in a very large number of services and composite services. These services could be driving a large number of other services, business processes and even applications. In the future the resulting plethora of services from the factory shop floor and other business level components will become difficult to manage and hence would require some sort of web service enabled composition technologies like Business Process Management (BPM) (Chen and Lu, 2009, Min-Jeong et al., 2007), Web Mashups (Guinard and Trifa, 2009b, Merrill, 2006), Open Platform Communication Unified Architecture (OPC UA) (Pramudianto et al., 2013), or Enterprise Service Bus (ESB) (Apache, 2012d, Fuse Team, 2010, Tu Quach et al., 2010) to manage them efficiently and effectively.

At the same time the Web Services Business Process Execution Language (WS-BPEL) (OASIS, 2007) has become the de-facto standard for orchestrating complex web services through invoking the operations of some existing web services. However the dynamic nature of the services and devices in the IoT environment can cause a considerable amount of state changes. In a plastics industrial environment the most interchanged messages are based on

some kind of event, alarm or notification. An example of this could be an alarm associated with the cavity pressure sensor or the notification of the piston position. These events are associated with an event source and one or more event consumers that react to the event according to a predetermined setting. The software architecture for this type of event driven communication and integration of systems is known as Event-Driven Architecture (EDA) (Minguez et al., 2011). The main features of the EDA are the asynchronous communication and interest-based message delivery using the publish/subscribe scheme. The timely dissemination of business-relevant events is of great importance in supply chain networks. For example the plastics material suppliers would be required to meet strict delivery deadlines when delivering materials to a μIM manufacturing process. In these situations it would be of great benefit for the enterprise to integrate the business workflows with manufacturing processes in order to meet the required response times and flexibility when reacting to manufacturing events (Minguez et al., 2011). This could be achieved by amalgamating the EDA and SOA through the integration of complex events into business processes. In this research we selected to use the ESB as it merges both the EDA and SOA paradigms for the purpose of integrating and linking heterogeneous platforms and environments. The ESB acts as the mediation layer to allow disparate applications to communicate with each other.

1.2 Aims and Objectives of the Thesis

The aim of this thesis is to propose an SOA based assimilation architecture using WSN for the purpose of integrating disparate industrial environment processes and business processes of an enterprise in particular the plastics machinery environment. The ESB has been proposed as the composition method for integrating the backend hardware and simulated machine process systems with frontend web visualisation applications. Furthermore web service based business processes are proposed to manage and orchestrate the resulting services from the architecture.

In this thesis the key process parameters in the μIM process are investigated; a new WSN based μIM process monitoring system, LVWSM has been developed for monitoring the μIM process. More specifically the ESB is proposed as the integration tool for a novel assimilation architecture, μIMP

Monitor. This architecture uses a WS-BPEL based business process framework as the glue to link disparate hardware platforms and to manage and orchestrate the resulting services from the various layers of the µIMP Monitor architecture. The key benefits of the proposed architecture are the reduction of complexity when integrating disparate hardware platforms as well as allowing the µIM process to be monitored on the web. In order to achieve this aim, the following objectives are pursued:

- To investigate the feasibility of monitoring the plastics machinery environment using WSN by:
  - Examining the challenges and issues associated with the use of WSN in monitoring industrial environments and processes.
  - Investigating the most common parameters used for monitoring the plastics machinery environment and machines.
  - Selecting a WSN for monitoring the plastics machinery environment by investigating the usability and reliability of WSN kits using PER, RSSI and LQI parameters.
- To develop a new µIM process monitoring system using WSN and the SOA paradigm.
  - Which can detect new sensor nodes when they are powered ON
  - Which can monitor the environmental and machine sensors
  - Which can present the sensor data for storage in a relational database and file format as well as being converted into a web service
  - That can communicate with the µIM machine sensors by designing a hardware interface.
- To propose hardware assimilation architecture for the purpose of integrating high-resolution µIM machine process data and low-resolution WSN (environmental and business process) data.
  - To develop a hardware middleware platform using the Enterprise Service Bus (ESB).
  - To develop process data simulation software for testing the integration architecture.

- To propose a Web Services Business Process Execution Language (WS-BPEL) based framework for linking the simulated process data with the environmental process data.
- To present the disparate processes data under a unified web portal by developing mini web applications using the Google Gadgets API.
- To integrate the ESB based assimilation architecture with the WS-BPEL framework and Google Gadgets based web portal as one complete test-bed for the purpose of linking and testing the disparate hardware systems.

1.3 Thesis Original Contributions

This thesis contributes to defining a hardware integration architecture, based on the SOA paradigm that can be used as a foundation for integrating disparate hardware platforms. The key original contributions of this thesis in the design and development of the µIMP Monitor architecture are summarized as follows:

- An investigation into the most common parameters used in monitoring the plastics machinery industrial environment was carried out, based on these parameters a comparative study of three WSN kits in terms of their usability and reliability in home/office and industrial environment settings was done.
- A comparative study of two WSN kits has been carried out in terms of their data communication reliability in home/office and plastics machinery industrial environment using the PER, RSSI and LQI parameters.
- The design and development of the novel LVWSM µIM process monitoring system that uses a WSN to abstract the functionality of the µIM process and environment and follows the principles of SOA by exposing these functionalities as Web Services.
- The design and development of the µIMP Monitor four layer architecture based on the ESB as the enabling middleware technology for implementing the SOA hence removing the integration complexity from the hardware systems.

- The design and development of GGs based mini applications for the purpose of monitoring the various sensor based processes under a unified web portal.
- The extension of the µIMP Monitor architecture by proposing the design of a new integration framework using sensor technology and business processes.
- A complete test bed has been developed which links the µIMP Monitor architecture with the WS-BPEL business processes framework and presents the backend hardware platforms and applications under one unified web portal using GGs.

1.4 Outline of the thesis

The rest of the thesis and remaining chapters are organized as follow:

Chapter Two presents a comprehensive literature review on the issues and challenges associated with the application of wireless technology and in particular WSN in industrial environments. An overview of the plastics machinery industrial environment in particular the µIM environment is given. The use of web technologies in particular the SOA and Web Services to allow WSN and other devices on the factory shop floor to be integrated with Web was investigated. The chapter pinpoints that the ESB which is the back bone of the SOA has been used in proprietary Enterprise-IT systems to allow interoperability but as of yet has not been applied to allow devices on the factory shop floor to interoperate. ***Chapter Three*** first identifies the most common parameters in monitoring and defining the plastics machine operating environment. Based on these parameters, a two stage selection criterion is used to select the best three kits for monitoring a typical plastics industry environment. ***Chapter Four*** evaluates the two remaining WSN kits in terms of their communication reliability using the PER, RSSI and LQI parameters in two distinct environments, the home/office and plastics machinery industrial environments. From the results of these experiments one WSN is selected for monitoring a typical plastics industry environment. ***Chapter Five*** presents the design and development of the LVWSM µIM system hardware interface and software. The LVWSM system uses the WSN selected in the previous chapter.

This system is used to monitor the environmental and μIM machine sensors in the plastics machinery environment. The test results using the LVWSM system show that the WSN is ideal for monitoring low-resolution data such as the environmental parameters but not suitable for the high-resolution machine data hence the need for a new architecture which allows the monitoring of low and high-resolution data. ***Chapter Six*** presents the design and development of novel distributed system architecture the μIMP Monitor, based on the middleware layer approach implemented using the ESB at its core. This architecture allows disparate hardware platforms including low-resolution WSN based platforms to be combined with high-resolution data platforms such as the National instruments (NI) Compact-RIO data acquisition system. In addition the chapter also presents the design and development of Google Gadgets based mini applications and process simulation applications to be used to test the architecture. ***Chapter Seven*** first presents the proposed integration framework in which the functionality of each stage in the μIM process is abstracted through the use of relevant sensor technology and web services and the integration with other standard business processes of the enterprise is done using the ESB and WS-BPEL. This updated architecture utilises business processes to allow the linking of the disparate hardware and software using WS-BPEL. ***Chapter Eight*** draws the overall conclusions and discusses a set of possible future activities from various research directions

1.5 Publications from Thesis

Raza, U.; Whiteside, B.R.Whiteside; Fun Hu; , "An enterprise service bus (ESB) and Google Gadgets based micro-injection moulding process monitoring system," Wireless Sensor Systems (WSS 2012), IET Conference on , vol., no., pp.1-6, 18-19 June 2012.

U. Raza, B. R. Whiteside, et al. "A Service Orientated Architecture enabled Wireless Sensor Network approach applied to measurement and control of a Micro Injection Moulding Process", PPS-27, Marrakech, Morocco (2011).

U.Raza, B.R.Whiteside, Fun Hu “A Service Orientated Architecture and Wireless Sensor Network approach using Business Processes applied to measurement and control of a Micro Injection Moulding Process Data.”PPS-29, Nuremburg, Germany, 15-19 July 2013.

U.Raza, B.R.Whiteside, Fun Hu “A Data Reliability study using PER and RSSI in plastics machinery environment.” A journal paper submitted to the IET Wirless Sensor Systems Journal, Under review, 2015.

U.Raza, B.R.Whiteside, Fun Hu “A Business Process Framework using Wireless Sensor Networks for the measurement and control of a Micro Injection Moulding Process Data.” A journal paper submitted to the International Journal of Control and Automation (IJCA), Under review, 2015.

Chapter 2: Literature Review

2.1 Background

There are a number of different types of networks used in wireless communication and these are defined by size and location. They include Wireless Personal Area Networks (WPAN), which cover small areas such as a home or personal workspace, Wireless Local Area Networks (WLAN), which cover small areas such as a university campus and Wireless Wide Area Networks (WWAN) which cover large areas; such as cities, (using satellites, telephone lines and radio waves to transfer data).

The development of VLSI technology and MEMS (Micro-Electro-Mechanical Systems) has made it possible to integrate traditional sensors, processors and wireless communication technologies to form low-power, low-cost, multifunctional sensor devices which can communicate untethered over relatively short distances. Wireless Sensor Networks (WSNs) do not need centralized infrastructure therefore making them; ad-hoc and be part of the network temporarily.

2.2 Wireless Networks for Industrial Environments

The competitive nature of today's industrial environments requires a complete overhaul to stay apace with the manufacturing markets. Industries have to improve both product quality and efficiency of their; systems, processes, and equipment. The equipment has to be reliable and available at all times with minimal maintenance in order to reduce operational and support costs (U.S. Department of Energy, 2002). Therefore, there is a requirement for low-cost industrial automation systems to; improve production performance, safety and to increase the operational life.

Industrial automation systems on the shop floor use wired communication networking technologies based on Fieldbus systems.. These provide the wired link between the programmable logic controllers (PLC) and hardware to form a process control chain. In these industrial systems, thousands of sensors may be connected using separate twisted shielded pair cable to relay gathered information to the controllers. Wired industrial networks

are very reliable but they do have their downside of being very costly. Temporary installation usually requires some process data before permanent installation. In some cases wiring can be difficult right from the start (e.g. rotating equipment), where too many measuring points, long distances between sensor and controller or hazardous environments are issues. Isolation of the cables can be critical, where cables are running near high vibration environments, high temperatures, magnetic fields or high humidity areas (Low et al., 2005). Maintenance costs can be high in difficult to access areas, where cable installation (e.g. large building structures) becomes faulty or connectors are damaged. For these reasons wired automation systems have their limitations,

2.3 Why go Wireless?

To overcome problems identified with wired industrial networks, companies are looking into ways, in which data can be; processed efficiently communicated over long distances to a large number of distributed locations, and from inaccessible locations. To achieve this, industry is turning to wireless solutions composed of distributed intelligent nodes that communicate data using electromagnetic waves.

Wireless technology can eliminate tens of thousands of feet of wiring from an industrial site and reduce maintenance costs (due to lack of replacement, inspection, testing, troubleshooting and repairing of old or faulty wires). Advancements in VLSI technology and MEMS (Micro-Electro-Mechanical Systems) has led to the convergence of traditional sensors, processors and wireless communication technologies, benefiting from; low-power, low-cost, multifunctional sensors and untethered communication over relatively short distances. These developments have influenced the idea of Wireless Sensor Networks (WSN). WSNs offer an expanding wealth of sensing capabilities and have self-configuring, self-calibrating, self-verification (Takruri et al., 2009). WSNs are also self-organizing and can rapidly organize and configure themselves into an effective communications network covering a large distributed area, enabling a range of production scenarios (U.S. Department of Energy, 2002).

2.4 Wireless Technologies for Industrial Environments

To communicate data without wires would require either wireless data orientated networks like WLANS and WPANs, or to add wireless capability to the existing wired industrial networks (e.g. Fieldbus).

This section provides a brief overview of wireless technologies which have specifically been designed or modified for industrial communications, specifically focusing on WLANs, WPANs and Wireless extensions of common bus technologies.

2.4.1 WLAN Technologies in an Industrial Environment

WLAN technologies process high data rates (megabits per second) over ranges of tens to hundreds of meters hence providing the user with untethered access to the Ethernet. Commonly used 802.11 wireless communications (Wi-Fi) has been adopted in many back-office IT infrastructures to extend the existing Ethernet network access to users outside of their wired environment. However, it has not been commonly used for industrial controls applications. Wi-Fi technology is becoming more common for long-range point-to-point communications such as supervisory control (e.g. offshore drilling operations). Wi-Fi uses public band frequencies (900 MHz and 2.4 GHz) and can possibly use proprietary frequencies to increase resistance to interference from other RF applications and increased security. However in ranges from one mile to over twenty miles, reliability due to environmental variations and other interferences (Hanssmann et al., 2009) becomes an issue.

Wired solutions based on fieldbuses and industrial Ethernet have so far been adopted to connect devices in industrial environments with varying complexity. The use of WLANs in industrial environments is stunted due to intrinsically non-deterministic nature and the deemed inadequate feedback control or critical safety. WLANs can be adopted in where timing requirements of control applications are not so tight. The author (Cena et al., 2007a) experimentally evaluates the statistical distribution of response times over a wireless channel when the characteristics of the background traffic are varied. The authors conclude that when 802.11 based devices can be used successfully whenever timing requirements are not so tight and even in these cases, however, care has to be taken so as to avoid excessive latencies. In

order to make WLANs suitable for real-time communications (e.g., multimedia traffic), the 802.11e specification (IEEE Std 802.11e, 2005) was introduced in 2005. This specification was analysed while in use at shop floor level in an industrial environment (Cena et al., 2008a). The authors concluded that despite the lower determinism and reliability, the resulting performance resembles closely to that which is achievable with fieldbuses. Delays are important issues for soft real-time applications and this paper carries out an analysis of delays in traffic patterns subjected to burst background traffic and its effect on the performance of real 802.11g and 802.11e networks (Zunino, 2007).

2.4.2 WPAN Technologies in an Industrial Environment

WPANs have; small size and form factor, short-range transmission (up to 50 feet/15 meters), low-cost and low-power consumption and are heterogeneous. They are known for ease of use (automatic synchronisation when the device is within range of the WPAN). They are designed to link personal devices together without cables and are intended for mobility and simple point-to-point communications. WPANs operate in the unlicensed Industrial Scientific and Medical frequency band at 2.4 GHz, with interference immunity, frequency-hopping capabilities and standardized interfaces.

2.4.2.1 *Bluetooth and HR PANs*

Bluetooth (BT) devices sharing the same frequency-hopping sequence can form a network called piconet. A BT device can play either the roles of a master or a slave. The master unit coordinates the traffic to and from up to seven active slave units. In a single piconet the slave units can only communicate with each other via the master. Each BT unit can be a member of up to four different piconets simultaneously (though it can be master in only one of them). A formation in which several piconets are interlinked in such a manner is called a scatternet. BT offers a gross bit rate of 1 Mb/s.

Studies into the application of HR-WPANs in an industrial environment are at an early stage and most of the studies use simulation methods to analyse different aspects of the HR-WPANs. A simulation study of the performance evaluation of video transmission over Ultra-Wide Band (UWB) WPAN is given by (Aripin et al., 2009). The simulation study by (Yasmeen et al., 2008) addresses scattering effect while modelling UWB channel in an indoor industrial

environment. The performance of UWB channel model is analysed following the parameters, such as power delay profile and the temporal dispersion properties. (Kuhn et al., 2008) present an accurately simulated complex UWB localization system in a dense indoor environment like a factory. This kind of simulation study could be a powerful tool to speed up the development time and reduce the cost for building such a system.

2.4.2.2 *LR-WPANs IEEE 802.15.4*

The goal of the IEEE 802.15.4 LR-WPAN (IEEE Std 802.15.4, 2006) standard was to provide short-range wireless connections at a very low cost and power. Data rates between 20 to 250 kb/s are achievable, with low data rates achieved when operating in the unlicensed 868/915 MHz band whilst high data rates achieved when operating in the 2.4-GHz cantered ISM band. This standard uses two network topologies depending on the applications requirements: the star topology or the peer-to-peer topology. A network coordinator device has to initiate, coordinate, and route communications on the network.

The study by (Feng et al., 2009) focused on real-time capabilities and reliability of IEEE 802.15.4 based WSN in an industrial environment. Both simulation environment and analytical methods were used to test against a benchmark. The study identified specific protocol limitations that prevent its application to delay bounded real-time applications. The authors propose some protocol modifications that enable real-time operation based on standard IEEE 802.15.4 compliant sensor hardware In a study evaluating IEEE 802.15.4 temporal performances in real-time traffic, heavy limitations were found based on cycle duration and device numbers and highlighted difficulties in heavy time constrained applications (Salles et al., 2008). The authors summarise that products, based on this standard, were found to add more restrictions and therefore the use of such technology for control command applications should us difficult. An investigation of interference in wireless sensor nodes allowed for setup optimization during interference (Bertocco et al., 2008) The authors aim was to measure parameters such as alarm latency, polling roundtrip time, number of failed polling's, and experimental cycle time to recognise interferences effects. (Lo Bello and Toscano, 2009)), provide a general methodology of cross-channel interference in co-located IEEE 802.15.4 networks in an industrial environment.

A number of industrial standards have emerged which build upon the IEEE 802.15.4 (IEEE Std 802.15.4, 2006) standards by developing the upper layers of the OSI reference model. These include ZigBee (ZigBee Alliance, 2006), WirelessHart (HART Foundation, 2009), and ISAI00 (Caro, 2008).

ZigBee: The ZigBee specification is built upon the IEEE 802.15.4 standard and is targeted at industrial control and monitoring, building and home automation, embedded sensing, and energy system automation. The IEEE 802.15.4 defines the PHY and MAC layers whereas ZigBee goes on to complete the standard by adding two high level layers; the network layer (NWK) and the application layer (APL) to the underlying structure. ZigBee has extremely low energy consumption and support for several different topologies. Later updates to ZigBee offered larger network capability, enhanced security features, and ability to change frequency channels when faced with noise and interference.

Wireless HART: Wireless HART is an extension of the HART protocol (HART Foundation, 2009), which is designed for process monitoring and control. It uses the IEEE 802.15.4 PHY layer with a modified MAC layer and implements frequency hopping, redundant data paths, and retries mechanisms in a mesh network topology. In the mesh networking topology each device is able to transmit its own data as well as relay information from other devices in the network (HART Foundation, 2009).

ISA100: The international society of automation (ISA) studies technical problems and develops standards for automation (Caro, 2008). The ISA100 focuses on wireless communication systems for monitoring and control applications, and approved a specification for process automation called ISA100.11 (ISA, 2009). The target of ISA100.11a is robust and secure communication in process automation. The ISA100.11a has wide application coverage, and provides the connectivity with different kind of networks. It will provide wireless connectivity for fixed, portable and moving devices for noncritical monitoring and control applications. The ISA100.1Ia will also build upon the IEEE 802.15.4 PHY layer with a modified MAC layer to provide a frequency hopping, multi-hop mesh network capable of inter-network routing (ISA, 2009).

IPV6 over Low-Power WPAN (6LoWPAN): In industrial settings, there is a requirement for connectivity between the wireless sensor network and the Internet hence solving many remote control and monitoring problems. The resource limitations of sensor nodes have been said to be too much for classical IP stacks (Mazzer and Tourancheau, 2009). To overcome this Internet Engineering Task Force (IETF) a new working group was established called 6LoWPAN based on IPv6. This protocol has all the capabilities of IPv6 on resource constrained nodes hence opening up the possibility of connecting to the internet. The Wireless HART and ISA100.11a standards also include 6LowPAN technology hence providing IPv6 connectivity to every wireless field device (Xin and Wei, 2008). The 6LoWPAN facilitates IPv6 connectivity over 802.15.4 compliant resource constrained devices by compressing the IPv6 packets.

In the 6LoWPAN protocol in order to realize the networking of devices and the interoperability of different equipment, it is necessary to institute uniform standard of network layer. Due to the address space and openness of grand scale, 6LowPAN (IPv6 over Low-rate WPAN) working group was formally established by IETF to institute LR-WPAN standard based on IPv6, the purpose of this group is that introducing IPv6 into LR-WPAN which takes IEEE 802.15.4 as its basic bottom layer standard.

2.4.3 Wireless Extension of Common Bus Technologies

Although extremely important in an industrial setting, wireless networks cannot be seen as a total replacement for wired networks in the short to mid-term period in the future. A strong focus on integration of wireless technology with popular wired industrial solutions provides a middle ground. The majority of industrial automation still use wired communication networking technologies based on Fieldbus systems (Mahalik, 2003, Pleinevaux and Decotignie, 1988, Zurawski, 2004) like PROFIBUS (Jecht et al., 2004), WorldFIP (Thomesse, 2004), or CAN (Hanxing and Jun, 2009) and Real-Time Ethernet networks (Decotignie, 2005) like ProfiNet IO (Profibus & Profinet international, 2005), Ethernet IP (ODVA, 2007) and Ethernet Powerlink (EPSG, 2003).

2.4.3.1 *Wireless Extension of Fieldbus Systems*

Early research on wireless fieldbus was conducted by the European Union OLCHFA project (Roberts, 1993), to provide wireless spread-spectrum transmission for the WorldFIP (Thomesse, 2004) formerly just FIP fieldbus. In this research it was shown that the wireless extension had the added advantage of microwave data transfer, maintaining high speed time-critical communication while still providing the obvious advantages over a cabled system such as cost and ease of installation. The disadvantage of this setup is that the processing of data could only be carried out at easily accessible distributed locations. For hard to reach locations, remote processing of data was not possible due to the limited processing power.

The potential of integration of standardized wireless technologies like IEEE 802.11, BT, or IEEE 802.15.4 with wired technology has been studied, and reviewed (Willig et al., 2005). One research opportunity identified in the fields of wireless fieldbus systems and wireless industrial communications is to assess the many emerging wireless technologies (ultra-wideband, MIMO techniques, smart antennas, wireless adhoc, and sensor networks) from both a technological and a market perspective in terms of their potential use in industrial applications. The key issues identified from some of the latest research issues were achieving timely and successful transmission of packets over error-prone wireless channels. A key observation in terms of the use of WSNs was the joint and careful design of applications and cross-layer networking stack to improve real-time capabilities. Other studies which have presented wireless extensions of current field bus technologies include the study by (Haehniche and Rauchhaupt, 2000, Rauchhaupt, 2002), a wireless Fieldbus called the R-Fieldbus based on IEEE 802.11 wireless standard and the wired Profibus-DP was presented. Another study showing the wireless extension of Fieldbus technology based on Profibus was presented for mobile vehicles to handle materials in a warehouse (Suk et al., 2002).

The use of the IEEE 802.15.1 and 802.15.4 standards as an extension to the current wired fieldbus technologies has also been investigated. An IEEE 802.15.1 Bluetooth based Profibus-DP was investigated by (Miorandi and Vitturi, 2005) which used Bluetooth Logical Link Control and Adaptation Protocol (L2CAP) to implement wireless communication and a wireless Fieldbus

based on BT was proposed in (OMRON, 2002). The study in (Meier et al., 2007) shows the application of BT technology in a harsh industrial environment and in this study six BT wireless Sensor Actuator Interface (SAI) devices link to a BT gateway which is connected to a PLC using a wired PROFIBUS Fieldbus. A design and implementation of a wireless Fieldbus based on IEEE 802.15.4 MAC protocol has been proposed (Dong-Hyuk et al., 2006). In the study by (Tang et al., 2008) a ZigBee-based wireless extension of FOUNDATION Fieldbus is put forward, providing wireless communication services for automatic monitoring and control for process industry. Another study not related to industrial applications by (Sleman et al., 2009) has shown that control appliances in an automated home connected to wired Fieldbus or other home networks can be easily integrated with wireless Fieldbus components by extending the communication protocol using 6LoWPAN wireless sensor networks.

2.4.3.2 *Wireless Extensions of CAN Systems*

DeviceNet is a low-cost communication link connecting devices to a network. It was designed bearing in mind the option of having several sub-networks inter connected by means of routers. The DeviceNet specification (Feng-Li et al., 2001) is based on standard CAN with an additional application and physical layer specification. It has received considerable acceptance in manufacturing applications and there are currently commercially off the shelf units by OMRON (OMRON, 2002) which can be used to add wireless capability to DeviceNet. The study by (Cena et al., 2008b) presents some possible extension of the DeviceNet using WLANs. The studies by (Kutlu et al., 1996b, Kutlu et al., 1996a) investigate the performance analysis of the Wireless Medium Access Control (WVM4AC) protocol and the Remote Frame Medium Access Control (RFMAC) protocol for a Wireless Control Area Network (WCAN).

2.4.3.3 *Wireless Extensions of Real-Time Ethernet Systems*

Although the IEEE 802.11 cannot cope with the very tight time schedule imposed by Profinet IO, the study in (Cena et al., 2008b) presents potential solutions as well as discussing wireless extensions to Ethernet/IP networks. (Kjellsson et al., 2009) propose amendments to WISA, which will improve the

802.11b/g coexistence and harmonize the integration of WISA for sensor/actuator communication in PROFINET IO. A number of Profinet IO devices have been implemented on a WLAN and the Profinet IO controller relies on wired connections (Petersen et al., 2008). Wireless extension of Ethernet Powerlink is implemented by means of the IEEE 802.11 WLAN in (Seno et al., 2009).

2.4.3.4 *Wireless Extensions vs WSN*

The modern factory shop floors require the latest automation features such as more flexible production and shorter product life cycles. To achieve these important features, industrial communication networks are required to support new types of traffic; be able to cope with the reconfiguration of plant layouts and computing resources in an efficient manner; be able to access the machines and equipment for diagnosis or programming purposes remotely and be able to globally access the overall information. Researchers are looking into wireless solutions to deal with these requirements of the factory of the future since they are the most effective option for re-cabling, installing and interconnecting new computing devices in a manufacturing system.

The previous sub sections on the wireless extensions of common bus technologies have shown that there are a number of ongoing research efforts to integrate different wireless protocol based technologies into popular wired industrial solutions based on Fieldbus technologies. The wireless extension of fieldbus technologies have mainly focused on standardized wireless technologies like IEEE 802.11, BT, or IEEE 802.15.4. In these research efforts the main focus has been on the issue of real-time and reliable data communication as well as a number of other issues including; being able to deal with short but frequent messages, be robust against electromagnetic interferences, provide the similar overall performance comparable with that of existing wired systems, but also to support extended functionalities (such as mobility, coverage of hard to reach and hazardous environments), in situ processing of sensor data as well as maintaining the interoperability and compatibility with existent communication infrastructures (Rauchaupte). The main trend in the implementation of wireless technology in industrial environment has been to add wireless extensions to the existing wired common bus technologies. The advantages include lower cost; minimal disruption to

current production; implementation to connect previously inaccessible and hazardous areas; the ability to gradually move towards a full wireless implementation.

Although the use of wireless extensions to common bus technologies has it benefits, there are instances when WSN can be used on their own E.g. the continuous monitoring of key environmental parameters on the factory shop floor or (temperature, humidity, light, etc.) Other situations when where WSN would be advantageous are when in situ processing of sensor data is required. But this would be limited based on the specifications of the node (memory processing and storage). For high-resolution data acquired at high sampling rates, there could be potential problems such as data transmission rate, storage and processing. In this instance wireless extensions would be beneficial.

2.4.4 **Summary of Wireless Technologies in Industrial Environments**

To summarise, there have been numerous studies carried out in the past decade and beyond on the use and adaptation of wireless technologies in an industrial environment. Most of the work carried out has focused on the wireless extensions of existing wired network architectures (Cena, Valenzano et al. 2007; Cena, Valenzano et al. 2008). The IEEE 802.11 based WLANs have received some focus and have been adopted in a number of practical cases where the timing requirements of control applications are not so tight. In order to make WLANs suitable for real-time communications the 802.11e specification (IEEE Std 802.11e 2005) was introduced in 2005. WPANs based on the IEEE 802.15 working group have becoming increasingly popular and have been implemented successfully in home office environments using the BT IEEE 802.15.1 (IEEE Std 802.15.1 2005) standard. They are starting to become fundamental technologies in industrial settings. Whilst the studies into the application of HR-WPANs (IEEE Std 802.15.3 2003) in an industrial environment are at an early stage, there have been a number of recent studies into the application of the IEEE 802.15.4 LR-WPANs (IEEE Std 802.15.4 2006) in an industrial environment. This has resulted in the

emergence of a number of industrial standards based upon the IEEE 802.15.4 including ZigBee (ZigBee Alliance 2006), WirelessHart (HART Foundation 2009), and ISAI00 (ISA 2009). Sensors and actuators based on wireless interface and protocols using these standards provide a new paradigm for industrial automation as they have integrated sensing, processing, control and communication capabilities in a single tiny node with the ability to form a network called WSN (Zhuang, Goh et al. 2007). Regardless of which wireless technology is being deployed these technologies are more susceptible to the harsh industrial environmental changes than their wired counterparts. The next section will give an insight into the latest challenges and issues related to the deployment of WSNs in an industrial environment.

2.5 WSN Challenges and Issues in an Industrial Environment

Before looking at some of the core issues and challenges related to the implementation of WSN in an industrial environment, we need to identify what properties a wireless device must have in order to operate reliably in a harsh industrial setting. A wireless device should have the following properties to operate in an industrial environment (Petersen et al., 2008):

- Reliable operation within a restrictive environment, (overcoming radio noise and obstructions and information black out areas).
- Be able to implement complex network algorithms with strict real-time requirements.
- Have embedded hardware and software platforms and be easily integrated with existing IT solutions.
- Must be able to take into to consideration restricted size, shape, construction and certification to meet stringent requirements.
- Work with limited processing power, memory, storage, and battery consumption.
- Ability to implement simplified ad-hoc and multi-hop network.
- Implement self-configurable, dynamic and adaptive network routing protocols.
- Provide application reconfiguration when required.

- Provide services within a dynamically changing environment.
- Have fault tolerance and recovery (self-healing, robust and reliable) capability.

In order to design robust and reliable industrial wireless sensor networks and protocols a number of important issues and challenges are being investigated by the research community (Willig 2008; Gungor and Hancke 2009). In the next section some of the most important issues and challenges (by far not all) which affect this research and have been encountered by researchers will be identified. and will give a brief overview of the latest research activities for each.

2.5.1 Resource Constraints in an Industrial WSN Environment

A typical wireless sensor node is composed of four basic units: sensing, processing, transceiver and power. The processing unit comes with integrated processing, memory, I/O ports and peripherals which reduce the need for additional hardware, wiring, energy and circuit board space. The processing unit has limited memory and processing capabilities. These restricted capabilities are important in implementing a WSN in an industrial environment (Gungor and Hancke, 2009).

2.5.1.1 *Processing Constraints*

The processing unit is fundamental for making the sensor nodes intelligent and to perform any computations on the sensed quantity. The study by (Stojcev et al., 2009) has shown that most of the processing units are based on CMOS MCU fully static devices which operate from very low frequencies from 1 kHz up to 32 kHz, to a maximum speed from 1 MHz at 1.8 V DC up to 100 MHz at 5 V DC depending upon the technology used. Processors using 4-, 8-, 16-, and 32-bit data bus width are implemented in sensor nodes. On average the power consumption in active mode of operation varies from 3 mW up to 30 mW for 4 and 8 bit processors and in power down mode is about 10 μW. Latest sensor node technology use 16/32-bit processors with a power consumption in active mode >100 mW and are intended for high capacity data processing. The main constraints poised by the processing unit include limited processing power and energy efficient computation capability.

Limited Processing Power: Traditional processing units are based on

Complementary Metal Oxide Semiconductor (CMOS) Microcontroller Unit (MCU) fully static devices which operate from very low frequencies from 1 kHz up to 32 kHz, to a maximum speed from 1 MHz at 1.8 V DC up to 100 MHz at 5 V DC depending upon the technology used. Although the processing power is sufficient to carry out basic computations and network coordination it will not be able to cope with the requirements of high capacity data applications such as image processing. Advances in latest technology have seen the introduction of sensor nodes with larger processing power such as the Sun Microsystems SUNSPOT (Sun Labs, 2010). SUNSPOT sensor nodes have 180MHz 32-bit ARM920T core processor. This can be enough for number of high capacity data applications. But this does mean that the lifetime of the battery is reduced compared to other sensor nodes with lower processing power. With this problem in mind there have been a number of efforts to improve the processing capability of the sensor node whilst not compromising the node battery lifetime.

Energy Efficient Computation using FPGAs: As every input quantity sensed, computation performed and message exchanged requires energy, emphasis has been placed on energy conservation for increased sensor lifetime by several years (Stojcev et al., 2009). Low powered system on chip technology has been used to reduce the energy consumption in sensor nodes, (e.g. field programmable gate arrays (FPGAs)). FPGAs have the ability to be reconfigured in the field. It strikes an optimal balance between processing power, energy requirements, and flexibility. An FPGA framework, Reconfigurable Wireless Intelligent Sensor Networks (REWISE), uses reprogrammable logic to allow hardware errors to be fixed and dynamically reprogrammed in the field and therefore reducing dead power for unused circuits (Wilder et al., 2008). This was emphasised by (Tanaka et al., 2008) who created a prototype for preliminary evaluation of the proposed mechanism. Experimental results show that this mechanism reduces enough power of its sensor nodes to prolong the lifetime of nodes without decreasing the processing time. Another study used wireless vision sensor nodes, combining FPGA and microcontroller system on programmable chip (SOPC) architecture resulting in low cost, low power consumption and reconfigurable nodes, (Chao Hu et al., 2009). Further studies (Krasteva et al., 2008, Nahapetian et al., 2007, Zhihua et al., 2008) all using a combination of FPGA and microcontroller processors in their sensor nodes,

improved processing system with low power consumption

Energy Efficient Computation using Data Compression: A number of studies have focused on using compression techniques to reduce the number of computations. The study by (Lantow, 2009) looks at energy consumption between individual paths from sources of raw data and the processed data. All the sensor nodes in this path have energy constraints and the depletion of the energy of any involved node would disable the network. A technique using the distribution of data processing to different nodes in the network to minimize local peaks in energy consumption was studied. The research by (Kayaalp et al., 2009) introduces a novel linear programming framework to model sensor network lifetime after data reduction through compression. Independent optimization of three data compression and forwarding strategies show that neither data compression nor flow balancing can achieve the maximal sensor network lifetime. However prolonged network lifetime was possible when both data compression and load balancing were optimized. In a study using WSNs to monitor sea water quality, genetic algorithms and cluster analysis were used to process front-end sensor node data and by filtering only abnormal data for transmission produced enhanced data compression. The resultant dramatic drop in energy consumption could also be used in industrial settings (Zhou et al., 2009).

Although energy consumption in sensor nodes is not the main focus of this study, the aforementioned studies have focused on reducing computation reduction in order to reduce energy consumption. Some of these techniques can be useful in this study as data collected from the industrial environment will need to be processed efficiently in order to reduce the impact on memory and storage requirements.

2.5.1.2 *Memory Constraints*

One of the main constraints for each sensor node is the amount of memory required to operate the application. The applications are normally stored on the microcontroller memory which is organised into well-structured functional partitions. These include the Flash/PROM memory which stores the application's executable machine code instructions. The RAM memory is used to store dynamic such as the Stack and Heap runtime mechanisms used by

modern programming languages. Non-volatile memory like EEPROMs is the third type of storage used to store important specific to each sensor node such as security keys, node identification information such as MAC address. This information is normally retained during node resets or power failures. There are a number of challenges for researchers including how to structure the application firmware so it can operate within the low-memory constraints of the sensor nodes (Electronic Design, 2009). Most of the studies have focused on different data compression techniques to reduce the memory footprint. compression algorithms. (Marcelloni and Vecchio 2008) propose a simple lossless data compression algorithm particularly suited to the limited storage and computational resources in sensor nodes. The evaluation results obtained through compressing of temperature and relative humidity data show compression ratios of 66.99% and 67.33% for temperature and relative humidity datasets, despite a lower memory occupation and a less computational effort. (Rein, Lehmann et al. 2009) present a novel compression algorithm called wavelet image two-line (Wi2l) coder that is designed to fulfil the memory constraints of a typical wireless sensor node. The Wi2l coder is designed for low-complexity sensor networks which in the past were considered to be not sufficient for image processing. (Lopez-Gomez and Tejero-Calado 2009) have proposed a lightweight architecture for sensor nodes based on the IEEE 802.15.4 standard suitable for sensor actuator systems. By comparing with the ZigBee standard and not using certain attributes they have reduced the RAM size. Results have shown reduced memory use and significant energy saving.

In the aforementioned research efforts to reduce the memory footprint, researchers have focused on compression techniques and use of less data intensive protocols. In this project the memory constraints will need to be considered as the data from the industrial environment will need memory in order to process and store the data. Some of the above mentioned techniques could be used in order to process quicker and large amounts of data.

2.5.2 **Reliability in WSNs**

For WSN applications running in an industrial environment some specific tasks in the application might need to be finished within a certain time limit. These applications or tasks with time deadlines are usually referred to as “real-time”

applications/tasks. Real-time applications or tasks require real-time operating systems, programming languages and real-time communications (requiring real-time communication protocols). Traditional real-time systems calculate worst case execution time and this dampens the ability to tackle the dynamic nature of the sensed information. On the other hand WSN systems are based on nodes which are resource constrained and have to be left unattended in most applications. Therefore the difficulty in WSN design is making optimal use of the limited resources and real-time intake of dynamic information from their environment.

2.5.2.1 *Real-Time as a reliability issue*

For WSNs the reliability of the network means the ability of the network to ensure reliable data transmission whilst continuous change in the network structure (Hanssmann et al., 2009). In these applications in addition to the resources and real-time constraints the data reliability has to be insured.

Problems which are common in the internet like packet losses due to congestion and delays due to queuing can also affect WSNs (Willig, 2008). While most of the research on real-time and reliability refers to the reliable and timely delivery of data packets within a WSN this does not mean that real-time and reliability can be just associated with packet delivery. There are other aspects that should be looked at such as reliable coverage in terms of number of sensors being deployed or the accuracy of the information itself or the chances of certain phenomena being detected (Willig, 2008).

The sensor data is time sensitive and needs to be received by the sink reliably in a timely manner. Data delivered late i.e. with long latency due to processing or communication may be outdated and lead to wrong decisions (Gungor and Hancke, 2009). Due to the harsh industrial environments, unstable communication link and the node itself, data packet loss is common. Therefore in order to ensure energy efficiency and at the same time be real-time and reliable it is critical to reduce the number of lost packets in wireless sensor networks. Although it is a huge challenge to ensure just one aspect such as reliability or real-time behaviour on their own, the nature of industrial applications and other critical applications is such that these two requirements need to be met simultaneously. To this end there have been a number of

research efforts to design protocols which meet these requirements either individually or simultaneously. Due to the fact that there have been huge number of research efforts focusing on real-time WSN protocols and reliable data transmissions on their own, it would not be possible to present all the research efforts in this section. Therefore the focus of the following section will only be on the latest research efforts incorporating real-time and reliability requirements individually and simultaneously. As these efforts will have taken into account any previous research efforts related to these requirements.

2.5.2.2 *Packet Error Rate (PER) and Signal Strength for Reliability*

The Packet Error Rate (PER), expressed in percentages and calculated as the ratio between the total packets sent and the number of successful packets received (by monitoring the ACK received in response) is an important Quality of Service (QoS) parameter for wireless networks. The Receive Signal Strength Indicator (RSSI) measures the power present in an incoming signal and it is not linked to the "quality" of the signal. The RSSI value is an integer range roughly from -100 dBm to 0 dBm for IEEE 802.15.4 radios. The RSSI is a measure of dBm, which is ten times the logarithm of the ratio of the power (P) at the receiving end and the reference power (Pref) and is shown in the equation below.

$$RSSI = 10.log\left(\left(\frac{P_{RX}}{P_{ref}}\right)\right)(dBm)$$

Equation 2.1 RSSI Equation: is a measure of dBm which is ten times the logarithm of the ratio of the power (P_{RX}) at the receiving end and the reference power (P_{ref}).

The Link Quality Indicator (LQI) on the other hand is a metric of the current quality of the received signal and is not related to the actual signal strength, but the signal quality often is linked to signal strength.

The PER along with RSSI and LQI parameters have been used in a number of previous research studies. (Amini et al., 2011) used XBee and XBee Pro modules in a body network setting to study how the Received Signal Strength (RSS) and the Packet Reception Rate (PRR) vary as the communication distance and transmission power levels are changed. The authors use the experimental results to perform transmission power control with

high precision in order not to exceed a certain PER.

The PER has been used to study the co-existence of the IEEE 802.15.4 WSN with other wireless networks as well as the communication performance in area perturbed by common home radio interferences and obstacles. (Lavric et al., 2012) studied the coexistence of the WSN IEEE 802.15.4, the IEEE 802.11g wireless networks and the ad-hoc Bluetooth networks that function within the same 2.4 GHz band. The communication quality is assessed using the PER parameter. A practical approach is used by assessing the PER parameter at various separation distances between networks and by varying the retries of a packet for the 802.15.4 network. (Simek et al., 2011) studied the coexistence of LowPAN devices with a Wi-Fi equipped laptop, Wi-Fi router and microwave oven in a home environment. The performance evaluation was done by measuring the Packet Delivery Ratio (PDR) metric. Results from this study showed that LowPAN communication efficiency can be significantly affected by the presence of the home appliances in the environment.

(Subaashini et al., 2013) presented an experimental analysis of the impact of various obstacles on the ZigBee RF signal strength in a smart home environment. The PER, RSSI, and LQI parameters have been measured and analysed and the authors concluded that as long as the PER is less than 2%, the communication between the nodes is acceptable and considered reliable.

The RSSI, LQI and PER parameters have been used in a number of studies to assess the reliability of communication in different environments. (Petrova et al., 2006) examined the reliability of the communication with a real IEEE 802.15.4 hardware by measuring the RSSI, PER and the run lengths distribution both in indoor and outdoor environments. The results from these measurements were used to calibrate and improve an existing simulator. The authors also address the coexistence between IEEE 802.11 and IEEE 802.15.4 and measure the impact these two wireless technologies have on each other when operating concurrently and in range. (Guo et al., 2012) use the PER, RSSI and LQI to assess the reliability of wireless nodes in the presence of interference from IEEE 802.11 (Wi-Fi), Bluetooth, and microwave ovens. Results from this study show that Wi-Fi and microwave ovens cause significant increases in the PERs to an upward of 25% depending upon the distances among the receiver, transmitter, and interference source. The authors also point

out that channel on which the sensor signals are transmitted also has an effect on the PER, particularly when considering Wi-Fi as the interference source.

Other studies such as the one carried out by (Staniec, 2012) provide both qualitative and quantitative assessment of providing reliable transmission in a ZigBee network. A variably loaded reverberation chamber was used to measure the PER under multipath conditions ranging from an unloaded to an overloaded chamber case. The key parameter used in all the measurements was the number of allowed packet repetitions (retries). The author gave recommendations regarding the optimal use of retries and their impact on ZigBee performance under different multipath scenarios in the reverberation chamber.

The research by (Srinivasan, 2006) tries to evaluate whether LQI is a better indicator of link quality than RSSI by using the PER. Results from this evaluation show that RSSI for a given link has very small variation over time for a link. Results also indicate that when the RSSI is above the sensitivity threshold (about -87 dBm), the packet reception rate (PRR) is at least 85% (so a corresponding PER is less than 15%). The authors also point out that around this sensitivity threshold, the PRR is not correlated possibly due to variations in local interferences such as noise. The LQI, on the other hand, varies over a wider range over time for a given link. However, the mean LQI computed over many packets has a better correlation with PRR.

The aforementioned research efforts have shown that the PER, and RSSI have been used extensively to study the data reliability in various environments including industrial and home/office. To data there has been no comprehensive study to compare and evaluate the data reliability in the two disparate environments using WSN from different manufacturers. In this research WSN from different vendors will be tested and evaluated for their data reliability especially the industrial environment. A comparison between the industrial and home/office environments will be done to assess the effects of the dynamic nature of the industrial environment on the data reliability.

2.5.3 **WSNs In Polymer Environments**

Although WSNs have been used to monitor various industrial environments, one industrial environment which still has not had much exposure to WSNs is

the plastics machinery industry. Common processes used for forming plastic components include: Injection Moulding (IM), Micro Injection Moulding (μIM) and Extrusion. A feasibility study by (Flammini et al., 2009) of WSNs in plastic machinery tested a four-node network on an IM machine using a thermocouple as input to each node. The aim of this study was to suggest a solution able to provide a high data, despite the low cost of the sensor nodes. The authors tested a simple software protocol that works even when the nodes did not have a stable clock reference. The basic idea was to synchronize nodes in order to oversample quantities of interest; in this way it was possible to reconstruct the signal even when the radio frequency (RF) link vanished due to interference with other RF sources. To mitigate against this interference, the authors exploited channel diversity by using a known backup frequency channel when the main selected channel becomes busy, avoiding continuous transmissions or reconnections. Most other studies have researched into the actual manufacturing of sensor nodes using polymer materials or polymer based lithium batteries.

Other areas of research using WSN in plastics machinery environment have been mainly focusing on the machine condition and fault monitoring. The research by (Liqun et al., 2012) proposes a novel industrial WSN for industrial machine condition monitoring and fault diagnosis. In this research, the induction motor is taken as an example of monitored industrial equipment due to its wide use in various industrial processes. Motor stator current and vibration signals are measured for further processing and analysis. On-sensor node feature extraction and on-sensor fault diagnosis using neural networks are then investigated to address the tension between the higher system requirements of IWSNs and the resource-constrained characteristics of sensor nodes. The authors also measure the battery life time of a node whilst transmitting machine diagnosis data every hour.

The research by (Snatkin et al., 2013) looked at real time production monitoring systems as an alternative to manual data collection and captures most of the required production data without human intervention. The general objective of the study was to analyse these systems and to offer particular solutions for small and medium sized enterprises (SMEs). In this study WSN

was used to monitor a lathe machines front bearing temperature and machine utilisation. The overall aim of this study was to assess whether National Instruments (NI) or Defendec WSNs were suitable for this purpose.

In the research by (LIU et al., 2015) a ZigBee WSN based client server monitoring system is developed to realise the real-time monitoring of the injection machine equipment operating parameters. The machine operating parameters were sent in real-time using an improved wireless data acquisition hardware module. A client software system was used to map the parameters in a graphical format.

Apart from the research by (Flammini et al., 2009) which tries to suggest a solution able to provide a high data using low cost sensor nodes, all the other research efforts used WSNs to focus on the monitoring of particular aspects of a machine in the polymer environment. No other studies looked at the use of WSN to monitor the complete IM, μIM, or Extrusion processes. Although WSNs have certain issues such as low data rates, limited resources, in this research a number of WSNs will be tested to assess their feasibility to monitor low and high-resolution data found in a typical plastics machinery environment.

2.6 WSNs and the Future Internet

Research into “Internet of Things” focuses on integration of physical parameters with integral parts of the internet and computer networks. With a plethora of sensory data available scattered around the globe, there has been a need for homogeneity, e.g. Open Geospatial Consortium (OGC) which has standardized the way to integrate data using the Sensor Web Enablement (SWE) (OGC, 2010) standards. This initiative has made the “Sensor Web” possible through which applications and services can access all types of sensors over the internet. So far it has been seen that in most cases, the usefulness of these WSNs is to extract information from the environment they are deployed in. This information is then processed by some external logic to achieve some goal (Garcs-Erice et al., 2009).

As these applications are of a proprietary nature, the resultant research, has mostly, focused on the issues related to the sensor devices and their networking capabilities. For an industrial environment the research community

has focused on issues such as; QoS, resource constraints, real-time data processing and delivery, data and network reliability, network scalability and interference factors.

WSN uses different network protocols and proprietary hardware and software platforms which lacks in interoperability between platforms. The problems include; the lack of uniform operations, resource reallocation and resource sharing and standard representation. The tight coupling of resource (utilization and deployment) and specific location, application, and devices employed plays into interoperability between these platforms and the problem of sensory data fusion.

Early work on sensor network designs have been geared towards the use of these networks in specific single applications (Akyildiz et al., 2002). Advances in the latest research and technology have allowed WSNs to be used in a wide range of application domains (Low et al., 2005). In an industrial automation domain a number of processes could be running at any one time as part of an overall manufacturing process. Each of these processes could have specific sensory inputs which can be provided only by specific hardware platforms. With this in mind a strong trend has emerged in designing WSNs composed of heterogeneous sensor devices which could be used for a wide range of large and complex applications like Enterprise-IT systems. This latest trend does come with its own set of challenges which need to be addressed in order to realize the full potential of current WSNs.

2.6.1.1 *Enterprise-IT Systems*

Growth of information and communication technologies (ICT) has seen business environments increase in size, with inter-department information data flow required for; decision making, timely and efficient procurement of product parts, inventory management, accounting, human resources and distribution of goods and services. Therefore organizations felt the need for efficient information systems to improve competitiveness by cost reduction and better logistics. With this context in mind the late 1980s and the beginning of the 1990s saw new software systems known Enterprise Information Systems (EIS) or Enterprise Resource Planning (ERP) systems which were specifically targeted towards large and complex business organizations with multiple

departments. These systems were of a very complex, expensive, powerful and proprietary nature which required specialist consultants to tailor implement them based on company to company requirements (Hossain et al., 2002).

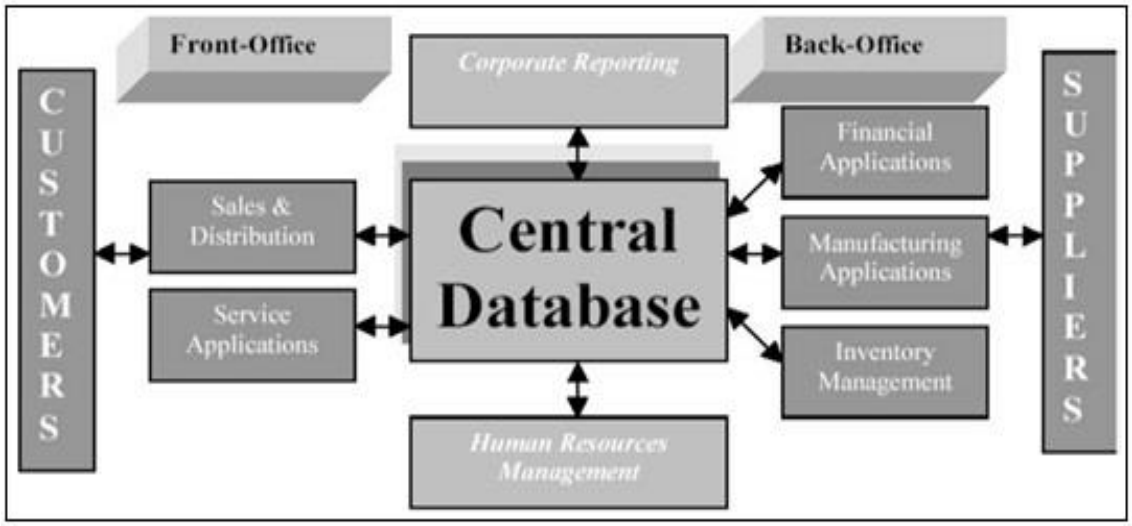


Figure 2.1 ERP systems concept (Davenport and Prusak, 1998).

ERP systems are systems for managing complex businesses with multiple departments. These systems are composed of modules supporting functional areas of each department such as planning, manufacturing, sales, marketing, distribution, accounting, financial, human resource management, project management, inventory management, service and maintenance, transportation and e-business. The ERP software architecture allows for the seamless integration of modules, providing flow of information between all functions within the enterprise in a consistent and transparent manner. Figure 2.1 illustrates the concept of an ERP system, with inter-functional data flow between the functional areas of an organization, which can be managed through an integrated ERP.

2.6.1.2 *ERP Systems and the Web*

The proliferation of the internet during the early to mid-1990s saw many of the major EIS vendors moving towards this technology. The aim of these organizations was to look for cost-effective ways to allow them to deploy the complex client/server based ERP technology beyond their organization's network using the Internet (Paulson, 2000). The adaptation of internet technology by ERP systems has allowed users or clients of all levels dispersed

around the globe to access information provided by the ERP system via simple web browsers.

The ability to link large amounts of complex information and business process environments across companies initially proved to be much more difficult than anticipated. This all started to change with the adaptation of Web Services into ERP systems. Web Services allow the integration of systems using XML messaging. According to its official description by (W3C, 2004), a web service is a software system designed to support interoperable interaction among computers over a network. Web Services are currently the preferred method for implementing systems that communicate with each other. Web services emerged during the early 2000s as a way to integrate systems using Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and the Universal Description, Discovery and Integration (UDDI) specification along with key XML technology. The Web Services approach helps to solve the challenge faced by many ERP systems of integrating business logic across diverse applications from different vendors.

A Web Service is the practical way of implementing services in a Service Orientated Architecture (SOA), which has become the leading integration and architecture framework. It allows software capabilities to be easily connected and reused hence making it quicker and cheaper to assemble, deploy and maintain enterprise technology.

Most EIS systems are built around the SOA architecture or have support for SOA web-based technology, which has seen the seamless integration of disparate information systems like Supply Chain Management (SCM) Manufacturing Resource Planning (MRP), and Financial Resource Management (FRM) from different vendors into ERPs. The factory shop floor is one area which still is at the early stages of implementing SOA. The future will see an increasing number of online smart factory shop floors built upon the SOA framework allowing them to be very easily integrated with existing ERP systems.

2.6.2 The SOA and WSN

In industrial environments, the use of robotics means that the processes are carried out with fewer workforces than ever before. Industries are looking into solutions to improve the quality of their products and improve the efficiency of their systems, processes and equipment. One such solution which has been the latest topic of research is looking at ways to integrate the dynamic harsh industrial environment of the factory shop floor with other business operations of the enterprise and external business partners through the internet hence allowing the remote monitoring of these environments. This would mean that the machinery and equipment as well as the business processes on the factory shop floor have to talk to other business departments and external partners in a common language. For this to happen it is crucial that WSNs are integrated with the internet and enterprise level applications. Therefore WSNs should be remotely accessible from the internet and, hence, need to be integrated with the Internet Protocol (IP) architecture (Gungor and Hancke, 2009).

The SOA is a promising technology, using a standardized interface, for integrating disparate systems within large organisations, enabling IT resources (e.g. departmental applications, embedded devices and WSNs), to be accessed as a service (Leguay et al., 2008). The SOA merges services and interfaces, by implementing standard protocols allowing access to information and business processes resulting in composite services with enhanced capability. SOA allows process visibility using standard interfaces without worrying about the service provider's technical details and hence promoting interoperability.

However if the SOA is applied to a real-time application based on WSN then the speed of the response required by such an application simply will not allow for an asynchronous communication model. For instance, WSN or response from a data acquisition system has to be synchronous, else the data received would be distorted and not in real time. The cost of implementing an asynchronous model would be higher, involving the use of reliable message queue software. Other implementations are possible as an asynchronous model is not always necessary, when a request-response or a publish-subscribe model is perfectly adequate, as is the case in many simple processes. This could be achieved through the use of an Enterprise Service Bus (ESB) which

supports various communication models. Furthermore, in this research the aim is to monitor the key parameters in an industrial environment and not to control them, therefore strict real-time guarantees are not as important.

2.6.2.1 Device Level SOA

The SOA paradigm can be applied to the individual sensor nodes at the device level or at the gateway level. At the device level the resource constraints (processing, memory, energy consumption, and storage) have been the main challenge in implementing the SOA. The Internet Protocol (IP) on sensor nodes is resource intensive (processing and memory), therefore sensor nodes are not considered suitable due to the low processing power, memory and energy consumption hence isolating them from the internet. So far the focus of the research for devices has been on existing web protocols and standards that allow a global network of computers to interoperate smoothly together despite of the many different software and hardware platforms available. The advantage of using web standards is that devices will be able to finally “speak" the same language as other resources on the internet, therefore making it very easy to integrate physical devices with any content on the web (Guinard and Trifa, 2009a).

One of the most common solutions has been through the use of the Device Profile Web Service (DPWS) a device level protocol. The authors (Savio and Karnouskos, 2008) have practically shown how the SOA can be realized at device level (PLC and SUNSPOT sensor node) through the use of the DPWS (Device Profile Web Service) a device level protocol. The research efforts by (Samaras et al., 2009, Sleman and Moeller, 2008) have implemented the DPWS protocol in a middleware on a gateway to allow the WSN to connect to IP based networks. Other device level implementations have included (Leguay et al., 2008) which proposes an SOA based protocols stack for WSNs called WSN-SOA and is used to bridge the gap between low capacity sensor nodes and high capacity nodes such as a web server. In all these efforts the low-resolution data is prevalent. Processing of low-resolution data is better carried out using RESTful (Fielding, 2000) Web services as these services are well suited for providing content to small footprint devices like WSN nodes. The investigations by (Guinard and Trifa, 2009a) have shown the possibility of

merging the WSNs with Enterprise Systems through the use of the light RESTful web services.

2.6.2.2 Gateway Level SOA

However, with high-resolution data which requires implementing services with complex inputs and outputs using WS*- Web services, the limited on-board capabilities at device level become rate limiting and a central gateway architecture or nodes with additional resources (processing, memory) are more suited (Guinard and Trifa, 2009a, Pautasso et al., 2008). By implementing the SOA at the gateway level every device and sensor on the shop floor will need to have its functionality exposed through the use of web services. This would give rise to issues similar to those faced by traditional SOA based enterprise IT systems and applications.

Different web service approaches have been applied to sensor node devices which convert the sensor nodes into a service. Regardless of the approach used, for example; a factory shop floor with a large number of sensor nodes, gives a large number of permutations services and composite services – hence the potential of multiple WSN installed for business operations in the future. A major deterrent to use has been the excessive and often blind overview, management requirements. This approach will come with its own set of issues and challenges such as service mediation, orchestration and management, scalability, sensor fusion, sensor data semantics, service governance and regulation to reflect business processes.

2.6.3 Web-based Composition Technologies

If every sensor on the sensor nodes in a factory shop floor is represented by a service, then this could potentially give rise to a huge number of services and composite services driving a large number of other services, business processes and even applications. The combination of these services from the factory shop floor and other services from the various business components of the EIS can result in a number of issues such as scalability, bandwidth, data routing, data fusion and presentation to name a few. The challenge of integrating and using these heterogeneous services to drive some meaningful processes across the various ES business components requires some sort of web services enabled composition technology which will enable the efficient

operation of these services with minimal load on the application and the network. (Delicato et al., 2010)

There are a number of composition technologies which create new services, processes or applications by integrating different sources on the Web. Some of the most common ones include Business Process Management (BPM), Web Mashups, and Enterprise Service Bus (ESB).

2.6.3.1 *Business Process Management (BPM)*

BPM can be seen as a business process composition technology used for the integration and management of business processes. BPM has been defined by (van der Aalst et al., 2003) as “supporting business processes using methods, techniques and software to design, enact, control and analyse operational processes involving humans, organizations, applications, documents and other sources of information.” Within BPM approach the business process is a “series or network of value-added activities, performed by their relevant roles or collaborators, to purposefully achieve the common business goal" (Ryan, 2009). The business process could be anything like the “purchase request” process in the purchasing department or it could be “machine specific part manufacturing” process in the manufacturing department.

The BPM approach sees these business processes as an essential part of an organization whose underlying functionality has to be managed, understood and enhanced in order to give the clients a better quality services and products. In BPM every operational process has a life-cycle which describes the various phases of the process. Figure 2.2 shows an overview of this life-cycle. The lifecycle starts at the **design** phase which focuses on existing processes or new processes. These processes are not limited to human-to-system workflows but it encompasses all other interactions including human-to-human. The focus at this stage is on all factors associated with the design process such as operating procedures, service level, the actors involved etc. The aim is to ensure an efficient design with minimal number of problems during the life time of the process. The design phase is followed by the **modelling** phase were the theoretical design is tested or simulated with different possible scenarios to see what would be the likely outcome before the actual implementation.

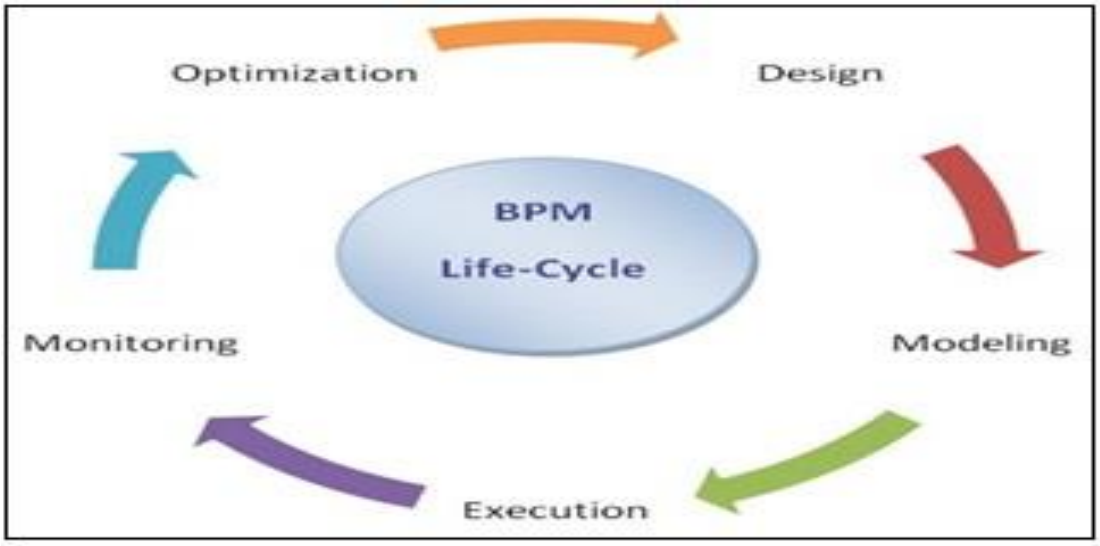


Figure 2.2 BPM Life-Cycle

After the modelling phase is the **execution** phase where the operational business processes are executed using the models of the theoretical design. The next phase is **monitoring** which keeps a track of the processes in order to look at how quickly they are being executed and if there are any problems. The statistics from the monitoring stage can be used to improve the process further. The last phase is of process **optimization** and in this stage all the data regarding the process performance from the modelling or monitoring phases is used to identify any actual or potential problems. Traditional BPM systems have been focusing on the operational business processes in an organization but this is all changing as every management thought needs the support of a technology. The SOA concept is one such technology which has been used to support the BPM approach due to its standards based, loosely coupled, and reusable services. While the BPM is used to improve the existing operational processes, the SOA can be applied to use its services to create new business processes as and when there is a new business need (Chen and Lu, 2009).

2.6.3.2 *Web Mashups*

A Web Mashup is a web application which can take information and functionality from different and possibly unrelated existing sources and combines them to create a new service or application. The Web Mashups can be seen as a web services enabled composition or integration technology. The

disparate sources can be RSS or Atom feeds, REST, JSON or SOAP web services, various XML formats, HTML, graphical elements, in the cases of SOA based WSN it could be sensor data or it could be simply another web page or website (Jin et al., 2008).

The application, content and presentation functionality can be integrated using publicly available APIs. The integration can be done in a number of ways for example, VB script JavaScript, in the browser and PHP, Java, C# for the server-side. Mashups offer a rapid application development process compared to other composition technologies. This is achieved through the use of Web 2.0 related technology. The architecture of a mashup application is made up of three components (Merrill, 2006): Content Provider, The Mashups Site, and the Web Browser.

Content Provider: This component is the provider of the information being mashed. The relevant data which needs to be extracted from the sources is exposed through standard web protocols like Web Services, RSS, REST, Atom etc.

The Mashup site: Is the place where the mashup logic is held. These can be implemented in the same way as traditional web applications using server-side dynamic content tools like PHP, CGI or ASP. The mashed information can also be made available in the clients Web browser using scripting languages like Java Script. The client-side mashup logic is normally composed of code embedded into the mashups Web page as well as API libraries from the scripting language.

The client Web browser: The browser allows the Mashup application to be graphically rendered and it is also the place where the client-side mashup logic runs. In some cases server-side logic is also used but this is for situations where the data composition might require more processing power.

2.6.3.3 *Open Platform Communication (OPC)*

The OPC interoperability standard was introduced to allow secure and reliable exchange of information between industrial automation space and in other industries. The OPC is a platform independent standard and guaranties the seamless exchange of data between devices from various vendors. The OPC Foundation is responsible for the development and maintenance of this

standard (OPCFoundation, 2012). The OPC standard was first released in 1996 with the aim of abstracting PLC specific protocols (Fieldbus technologies such as Modbus, Profibus, etc.) into standardized interfaces. This allowed systems like HMI/SCADA to be interfaced with other systems and devices via a “middle-man” which converted generic-OPC read/write requests into device-specific requests and vice-versa (OPCconnect, 2012). The implementation of this standard saw the emergence of a new breed of products which allowed end-users to implement systems that could interact seamlessly via OPC.

The OPC standard is a series of specifications which define the interface between clients and servers, servers and servers, and access to real-time data, monitoring of alarms and events, access to historical data and other applications. The OPC was originally designed to provide interoperability between windows based software applications and process control hardware. It defines consistent methods of shop floor data access regardless of the type and source of data. The methods provided to an OPC client by one OPC server for a specific hardware device are the same as any other OPC Server for that same and any other hardware device. Traditionally, when an application or software package required access to data from a device, a custom driver had to be written. The OPC solved this issue by defining a common interface which is written once and then reused by any SCADA/HMI system or custom software package which can act as an OPC client. The overall aim was to reduce the amount of duplicated effort required when interfacing devices (OPCFoundation, 2012).

OPC Unified Architecture (UA): The initial OPC specifications were restricted to the Windows operating system and used Microsoft’s OLE technology (Component Object Model, or COM) to communicate with clients and were known as OPC Classic. These specifications were adopted across multiple industries, including manufacturing, building automation, oil and gas, renewable energy and utilities, among others. The introduction of the SOA in manufacturing systems saw the emergence of new challenges in security and data modelling. To meet these challenges the OPC Foundation developed the OPC Unified Architecture (UA) specifications which provide open-platform architecture based on the SOA. The reasons for the OPC Foundations decision to propose a new architecture are summarized below (CAS, 2012):

- Microsoft has emphasised Web Services and SOA in favour of COM.
- OPC vendors want a single set of services to expose the OPC data models
- OPC Vendors want to implement OPC on non-Microsoft operating systems, including embedded devices
- Other collaborating organizations need a reliable, efficient way to move higher level structured data

The OPC UA is a set of specifications that define a common infrastructure model to facilitate information exchange in manufacturing systems and software. Recently, it has adopted web service standards to increase interoperability among business systems at the enterprise level. The latest version of OPC UA is completely based on the SOA.

The original OPC interoperability standard was introduced to allow exchange of information between industrial automation space and in other industries. But it had two main issues, Windows-platform-dependency of OPC and the DCOM issues when using remote communication with OPC. DCOM is difficult to configure, has very long and non-configurable timeouts, and cannot be used for internet communication. The OPC UA resolved this issue as it was cross platform and it supported a very wide range of protocols and formats. At the point of kit selection this technology was newly available, it had significant drawbacks. Many of these are practical aspects which are due to the novelty and the extent of the new specification. The major concern of the new technology was its performance. The main performance issues concern data transmission and computational requirements. The use of text based XML messages in transferring information used a substantial amount of the bandwidth from the information throughput viewpoint. The other performance issue identified was that the security module took a lot of the processor time.

But the main reason for not choosing the OPC UA for this project was that when it came to application development, the new specification gives additional workload. Because of the extent and elaborateness of the specification, plenty of work has to be done even when creating small applications (Leitner, 2006). Even though whole interface does not need to be implemented, the amount of work to achieve at least some minimal functionality is much greater than with the classic OPC. Finally, another issue is at the time OPC UA was a recent

specification and not as many products support OPC UA than the classic OPC.

2.6.3.4 *Enterprise Service Bus*

The Enterprise Service Bus (ESB) is an open framework which helps to compose, deploy and coordinate communications between service components of a distributed system. The ESB is an agile pluggable integration framework which can allow any new service to be easily plugged into the bus. The ESB acts as an abstraction layer over the messaging framework. The SOA is most commonly implemented using Web services although it can be implemented with all technologies that directly implement service interfaces using WSDL, and communicate with XML messages. So far the SOA has been implemented in small to medium size applications. However applications which need to run enterprise level logic using the SOA will find it difficult due to the high number of individual interactions required. Therefore a more complex and manageable setup is required which can support enterprise level services, integration with existing infrastructures and other established composition styles like message-orientated and event-driven integration.

There are two possible methods for integrating technology with information systems; firstly direct connection patterns with the build conforming to client modules and secondly inserting composition logic between the client and server modules (Mike and Willem-Jan, 2007). For the former, this will involve building interfaces for each connection resulting in a point-to-point topology. This form of composition topology is not scalable and becomes very hard to manage as the number of integration point's increase with the increase in the number of systems. For the second option, the composition layer must support interoperability among, and coexist with deployed infrastructure and applications. One such composition layer which can appropriately manage an integration framework for Web services and SOA whilst supporting the above requirements is the Enterprise Service Bus (ESB) (Keen et al., 2004).

ESB Architecture and Components: ESB standard based messaging framework is based on the bus architecture as shown in Figure 2.3. The ESB has been designed to provide interoperability between applications and other components via standards-based adapters and interfaces. The bus is used as the transport and transformation hub for the distribution of services across disparate and distributed systems and computing environments. The ESB

brings together applications and discrete integration components to create assemblies of services to form composite business processes, which in turn automate business functions in an enterprise.

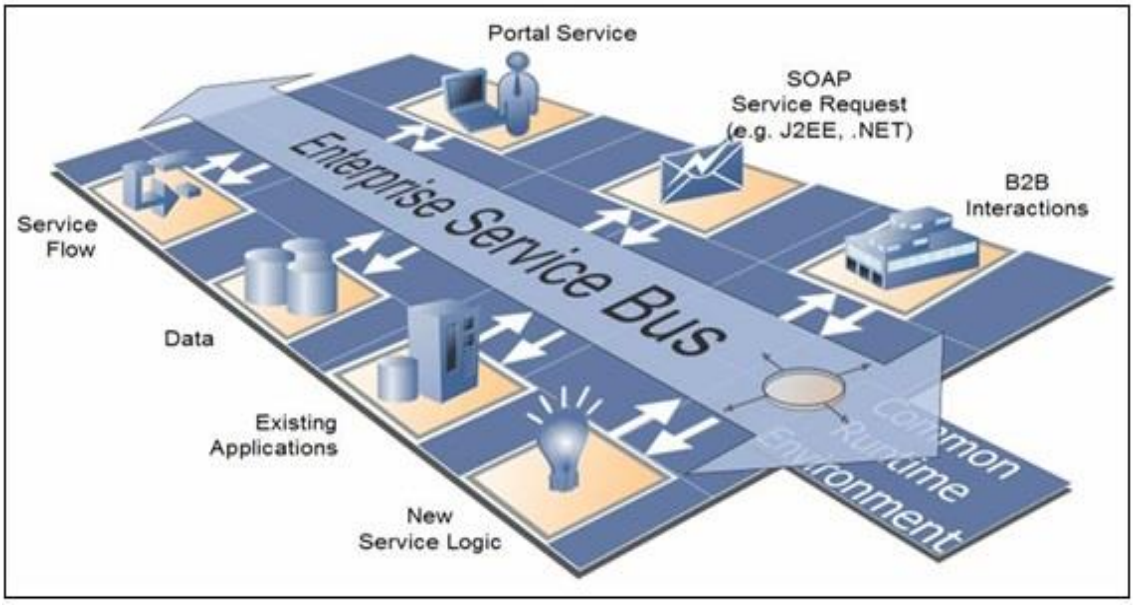


Figure 2.3 The Enterprise Service Bus (ESB) (Keen et al., 2004).

Some of the important features of the ESB messaging backbone are as follows (Doshi, 2009):

- Transformation of messages into different formats
- Protocol conversion
- Routing of messages
- Service and Event orchestration allowing seemingly unrelated sets of services and even entire applications to be choreographed to work together to define new business processes,
- Message acceptance and delivery from various services and applications linked to the ESB
- Service authentication and authorization, data integrity, encryption and auditing of all messages moving within and between ESB instances and other service-oriented applications.
- Provides auditing, logging & reporting facility.
- Administration and monitoring functions to find the faults and to monitor the performance of the ESB.
- Has a service registry that contains and manages the metadata

that describes the Services.

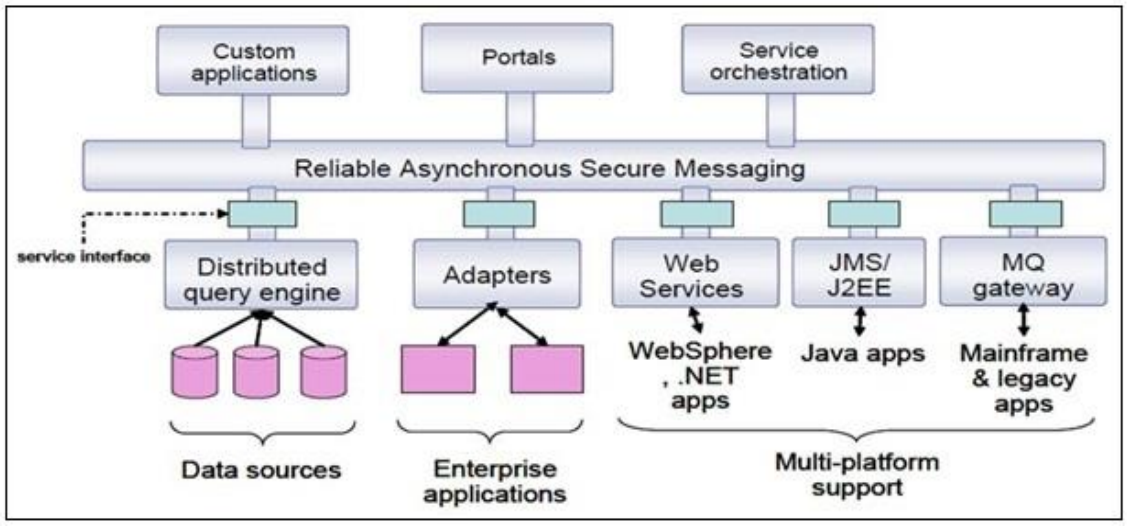


Figure 2.4 ESB connecting a range of applications and technologies (Mike and Willem-Jan, 2007).

The services use the ESB as a go-between with other services and don't interact directly with each other. The ESB acts as a message broker between the applications, making it a loosely coupled architecture. The ESB provides a variety of transport bindings for invoking services. A binding component (BC) acts like an interface to a remote service. Every different way of interacting with external services has a BC e.g. FTP, email, HTTP, JMS, SAP, RSS, XMPP, REST, JSON and SOAP. The ESB reduces the number of point to point connections hence reducing the issues linked to spaghetti like systems based on point-to-point topology (Doshi, 2009).

Asynchronous events, such as ad-hoc ordering of products leading to specific manufacturing, cannot be designed a-priori but must be defined dynamically. In this situation supporting enterprise applications communicate using an Event-Driven SOA (Doshi, 2009). In an ESB, applications and event-driven services are required to be linked together loosely, allowing them to operate independently from each other while still providing value to a broader business function. The distributed nature of ESB allows it to integrate such event-driven services across a variety of platforms, protocols, and technologies. It allows individual event-driven services to be plugged into the ESB backbone on an as needed basis, be highly decentralised and work together in a highly distributed fashion, while they are scaled independently from one another (Mike and Willem-Jan, 2007). Figure 2.4 shows such an example where applications

running on disparate platforms are abstractly decoupled from one another and can be connected through the bus architecture as logical endpoints that are exposed as event-driven services. The example shows the integration of a J2EE application using a JMS BC, a dot NET application using C# client, an MQ application that interfaces with Legacy applications as well as other external applications and data sources based on the SOA.

The endpoints in the ESB shown in Figure 2.4 provide abstraction of the physical destination and connection details. This is done by allowing the services to communicate using logical connection names which are mapped at runtime to actual physical network addresses hence allowing the services connected to the ESB to be upgraded, replaced or moved without the need for code modification and disruption to existing applications. To ensure a relatively fail-safe operation and guarantee communication link despite network failures and outages duplicate processes can be setup and endpoints can be configured to use several levels of QoS.

ESB and Industrial Environments: The integration of industrial processes on the factory shop floor with enterprise level applications requires the merger of the sensor data in the industrial domain with the SOA paradigm. The last few years has seen various research efforts looking into the application of the SOA to the industrial automation domains. The main focus has been through the use of WSN to extract the sensor data from the environment and then converting this data into web services to be used in the SOA (Savio and Karnouskos, 2008, Samaras et al., 2010, Pramudianto et al., 2013). The SOA paradigm has been applied to the individual sensor nodes at the device level or to the gateway middleware (Pautasso et al., 2008, Guinard and Trifa, 2009b). Other efforts have included the implementation of the SOA to networked industrial controllers such as PLCs (Candido et al., 2013, Jestratjew and Kwiecien, 2013, Pramudianto et al., 2013), Robots, conveyors (Starke et al., 2013) and industrial machines(Gilart-Iglesias et al., 2011, Macia-Perez et al., 2009). More lately the virtualisation of services is transforming the factory of the future into a "cloud of services", where dynamic resource allocation and interactions will take place (Karnouskos et al., 2012, Chenhui and Xinran, 2012).

Whatever the implementation scenario of the SOA, this results in a large

number of services and composite services driving a large number of other services, business processes and even applications. The resulting plethora of information extracted through the use of services can become difficult to manage and requires an optimal way to categorize and make it available to multiple systems. Therefore there arises a need for a composition technology such as the ESB to handle the complexity as well as be able to cope with the heterogeneous nature of the different devices used and allowing them to operate seamlessly under one platform. The ESB acts as the middleware layer which provides a mechanism for mediation to simplify the task of connecting distributed systems. A middleware in general can be seen as a layer between applications and operating systems which provides a simple, reliable way for integration in a distributed programming environment (Kotsopoulos et al., 2010).

The ESB has been used in many types of enterprise systems including healthcare (Schramm et al., 2012), marine data (Huang et al., 2010), education (Hongping et al., 2010, Chun et al., 2012), power management systems (Pingshun, 2011, Li et al., 2012), network management (Kotsopoulos, 2010), air quality monitoring (Tu Quach et al., 2010) to name a few. In all these applications the ESB was used for the purpose of allowing these systems or applications to be interoperable with other disparate IT systems. In industrial environments the ESB has been implemented mainly for logistics and supply chain information systems (Luyang et al., 2006, Zhan et al., 2010), product design and life-cycle (Silcher et al., 2010), material control (Min-Jeong et al., 2007) and energy systems (Pingshun, 2011, Silcher et al., 2010). The study by (Kwei-Jay and Panahi, 2010) has proposed a real-time SOA architecture using the Llama ESB. In this study the architecture was not implemented on the factory shop floor. A real-time Java based application was developed to test the reservation performance of a service being reserved in real-time. To the best of the author's knowledge there is only one study which is similar to this research (Garcs-Erice et al., 2009) which has looked at integrating different WSN platforms under one unified platform using the ESB. This study uses a publish/subscribe middleware to interconnect the WSNs and does not apply this to other equipment or machines in the industrial environment, It also does not use any orchestration engine to connect any business processes. Therefore there arises a need for an approach which utilises the ESB to handle the

complexity as well as be able to cope with the heterogeneous nature of the different platforms used and allowing them to operate seamlessly under one platform. As of yet there is no study that has looked at integrating and visualising different environmental data and machine data in an industrial environment using the ESB under a single unified platform to run integrated business processes. In the later chapters the focus of this thesis will be on such a novel system which will allow the integration, interoperability and visualisation of hardware based systems under a single unified web portal.

2.6.4 Business Processes

BPEL stands for Business Process Execution Language and is an XML-based workflow definition language which can be used to link sometimes disparate functions into an integrated process. BPEL allows businesses to describe inter or intra enterprise business processes that are connected via Web services. It becomes the glue to bind Web services into a cohesive business solution, facilitating their orchestrated interaction both within and between enterprises. With the use of BPEL a business process can create a number of web services hence allowing the creation of a completely new business application with its own public interface to end users (internal or external). BPEL opens a completely new way or at least enhanced way, for software development for mainstream business applications to allow a programmer to describe a business process that will take place across the Internet (Cobban, 2004).

2.6.4.1 *BPEL and WSNs*

Business process execution using WSNs is a new research field with only a few approaches. (Spiess et al., 2009b) introduce a business process partitioning approach to manage efficient business process execution in dynamic infrastructures. Part of this approach uses BPEL to map the business process as service calls. An earlier work by the same author uses BPEL to design a WSN application for monitoring hazardous goods. The authors do not describe, which part of the application is designed with BPEL and if it is used in the sensor network or only on the backend computers. (Glombitza et al., 2009b) presents an approach to integrate WSNs into SOA technologies such as XML, Web Services and BPEL using XML compression and a new transport protocol

for Web Services called Lean Transport Protocol (LTP). Although BPEL has been implemented on a server machine as it is not suited for resource constrained device, the author does plan to implement it directly onto the sensor nodes in the future. There have been earlier attempts to implement BPEL onto mobile devices such as PDAs and other embedded devices such as the work by (Hackmann et al., 2007) in which a lightweight BPEL engine is implemented which is small enough to be deployed on mobile devices that are able to run the reduced Java ME version of the Java VM. Other graphical tools similar to BPEL which use proprietary communication protocols include the GWELS (Graphical Workflow Execution Language for Sensor Networks), (Glombitza et al., 2009a) language and toolkit.

2.7 Conclusion

The trend to fully replace wired technology with wireless technology in industrial environments like factory shop floors is still limited due to the harsh radio-hostile environment and the costs involved. Industrial organisations are more inclined towards adding wireless capabilities through different wireless protocols (IEEE 804.15.4, ZigBee, WiFi, BT) to the existing wired fieldbus technologies as this approach is more viable and allows businesses to partially upgrade part of the infrastructure which were not accessible due to wiring constraints. Therefore allowing them to test and evaluate this relatively new technology at lower costs before investing into full scale replacement.

The SOA paradigm has been prospering in the Enterprise-IT systems and the internet for a long time. It is fast becoming the leading technology as a means for integrating disparate systems within large organizations. This architecture enables IT resources like specific departmental applications, business partners, business processes, systems and more lately even hardware resources like embedded devices such as sensor nodes. The factory shop floor is still one area which still is at the early stages of implementing SOA and researchers in the field of WSNs have started to adopt this architecture to tackle the above mentioned issues hence resulting in a number of research efforts and solutions.

From these research efforts it has become evident that the SOA paradigm can be applied to the individual sensor nodes at the device level or to

the gateway middleware. The device level implementation seems to be the better and faster approach but it does come with the resource constraint and energy efficiency challenges. Different web service approaches have been applied which convert the sensor nodes into a service. Regardless of the approach used, if applied to a factory shop floor environment were there can be a large number of sensor nodes then this can result in potentially a large number of services and composite services which can pose a huge challenge to manage as it is often difficult to keep an overview of all the devices and services. The challenge of integrating and using these heterogeneous services to drive some meaningful processes across the various enterprise business components requires some sort of web services enabled composition technology, which will enable the efficient operation of these services with minimal load on the application and the network.

In the next chapter the most common parameters in monitoring and defining the plastics machine operating environment will be defined. Based on these parameters, a number of WSN kits will be selected. According to the device specifications and data sheets of the selected kits, the most suitable WSN will be identified. To validate the choice, a number of experiments will be carried out to assess the suitability of the chosen kits for monitoring a typical plastics industry environment.

Chapter 3: Evaluation of WSNs for Usability and Reliability in Industrial and Home/Office Environments.

3.1 Introduction

Industrial environments like the factory shop floor, are typically composed of an array of machinery and ancillary equipment, there can be thousands of sensors, actuators and I/O components. The majority of these industrial setups are still connected using wired communication networking technologies based on Fieldbus systems. These networks are very reliable but they do have some fundamental shortcomings of being very costly and not covering inaccessible and hazardous areas due to the cabling constraints. In order to overcome these problems companies are looking into ways of collecting information using smart sensors and are turning to wireless industrial systems composed of Wireless Sensor Networks (WSNs), which are made up of distributed intelligent nodes that communicate data using electromagnetic waves as their transmission medium.

There are a number of WSN development kits available from various vendors and to date there is no comprehensive review of these kits being tested in industrial settings. To make up this shortfall, this chapter first identifies the most common parameters in monitoring and defining the plastics machine-operating environment. Based on these parameters, a number of WSN kits will be selected. According to the device specifications and data sheets of the selected ones, the most suitable WSN kit is identified. To validate the choice, a number of experiments were carried out to assess the suitability of the chosen kits for monitoring a typical plastics industry environment.

3.2 Identification of the Monitoring Parameters

In order to monitor the plastics industry environment some of the most common key parameters, which have been used in previous research, need to be identified. The Micro and Nano Technology (MNT) laboratory has IM and µIM machines mainly for IM and µIM industrial processes. From previous studies the most common environmental parameters monitored have been temperature and humidity (Westerdale et al., 2008, Islam and Hansen, 2009). At machine

level using IM, the most common process parameters monitored include temperature and pressure, in particular in the cavity. The µIM process on the other hand being fundamentally similar to the IM process comes with additional challenges related to the control of piston movement and melt storage barrel settings (Whiteside, 2006, Greener and Wimberger-Friedl, 2006). This makes the process setup and optimisation more difficult to perform compared with the conventional IM process. The micro dimensions of the mould and the subsequent components manufactured using this process also pose a significant challenge. The quality of these parts cannot be assessed without using visual techniques such as 3D optical microscopy, high speed cameras, micro-tomography, atomic force microscopy 3D scanning electron microscopy (SEM) etc. Therefore, additional monitoring of the key stages of the process can be beneficial in order to ensure the quality of the product being up to the standard. To summarise the plastic manufacturing environment requires monitoring of:

a) Temperature
b) Humidity

At the machine level, a µIM process requires the monitoring of:

a) Cavity Pressure (high-resolution data)
b) Nozzle Pressure (high-resolution data)
c) Temperature (high-resolution data)
d) Piston Displacement (high-resolution data)
e) Piston Velocity (high-resolution data)

The µIM process usually requires high specification monitoring due to the high pressures (>2000 Bar), high injection speeds (700 mm/s), and cavity filling times can be as low as 1ms. Therefore, an ideal system should use a 16-bit analogue to digital convertor with a sampling rate of 2 KHz and above, although higher sampling rates will give better results. Therefore the bandwidth required can be calculated as follows:

Bandwidth = no of channels (ideally 16) x rate (2KHz) x bits (16 bit) = 512 Kb/s.

With these parameters identified, a WSN kit needs to be selected which can meet the requirements of monitoring the above identified parameters.

3.2.1 Initial selection criteria of WSN

The selection criteria for the WSN need to take into account the monitoring parameters identified in the previous section. Therefore, a WSN node needs to have the following features to cater for the above parameters:

- **High Processing Power**

  In both the conventional and µIM processes the continuous monitoring of key process parameters such as injection piston dynamics, injection/cavity pressures and thermal properties has been pivotal for the overall process optimisation and control. For example, the responsiveness of the pressure and cavity measurements to process variation can be assessed by recording peak and integral values of the injection and cavity pressure curves. The µIM usually requires high specification monitoring systems due to the high pressures (>2000 Bar) and injection speeds (<10ms cavity filling times). With typical sampling rates being above 2 KHz means that there will be a large amount of data to be processed. This poses a challenge of separating the key data from such a large dataset. Therefore, it would be beneficial if the sensor node could have on-board high processing power to carry out this task and only send the key data to the controller node. Sampling rates of 2 KHz and above require high computational overheads for example using 16bit resolution and sampling time of 5 seconds would generate approximately 20Kbytes of data. If the sampling frequency is increased to 5 KHz then this would generate 50Kbytes of data every 5 seconds. On the other hand, too low a sampling rate leads to loss of data fidelity. If the process needs to be monitored continuously then this can generate large amount of data, which needs to be stored and processed. Therefore, to analyse data on this scale on-board a sensor node would ideally require a multi-core processor, which could dedicate some of the cores to process the data. You would also need to choose a sampling time such that the chance of data loss is minimised and at the same time leaving enough time for the microcontroller to process the data within two sampling instants.

- **Memory and Storage**

  The other challenge associated with large amounts of data is the memory and storage requirement. This goes hand-in-hand with the processing capability. The need to continuously monitor the key parameters of the process necessitates sampling for longer periods. This will inevitably result in large amounts of data. As an example if sampling at 2 KHz with a 12-bit resolution for 1 hour would result in approximately 11Mbytes of data. At such high sampling rates, the size of the data captured can easily increase to gigabyte and greater scales. Therefore, it is imperative that the data be processed in real-time and only the key data is kept to save storage space. The memory required to host the firmware also needs to be taken into consideration as well as the way the user application is written (avoid using long integer or double arrays etc.) making sure not to use too much memory. All the above necessitates a large on-board storage capability in the form of either flash memory or hard disk. Therefore a sensor node with on board flash memory of 8GB (storage) and above along 512 MB of RAM would be ideal.

- **Good transceiver data rate and range**

  In the Polymer MNT laboratory, the distance between the μIM machines and the controller node can vary between 5m to 10m. The material used in the manufacturing process is stored in a cupboard and a separate storage room. The cupboard is approximately 10m away whereas the storage room is more than 50 meters away. Therefore, in order to monitor the machines, material and the environment a sensor node is required with a transceiver, which can cover all the above ranges. The typical range for a ZigBee based wireless sensor node is 10 - 15m although longer ranges maybe possible depending upon the environment condition and clear line of site. This can theoretically increase to 100m if a high power node is used. When used in a network the range can be increased further to cover wider areas. Therefore, a sensor node using the ZigBee protocol in a WSN would be ideal to cover the Polymer MNT laboratory. The large amounts of data generated would need to be transmitted to the coordinator node. Therefore a transceiver that

supports high data rates in the range of 500kb/s and above would be ideal.

- **Wireless protocols support**

  The data stored on the sensor nodes needs to be transmitted to the controller node reliably without any errors. The Polymer MNT laboratory can become more of an industrial environment when all the machines and equipment are running at the same time. Depending upon the placement of the sensor nodes (machine, storage or environment) data reliability can be an issue. The harsh industrial radio-hostile nature of the environment and machines will have an effect on the data reliability. The ZigBee Alliance released the ZigBee PRO specification (ZigBee Alliance, 2007) in 2007 which is aimed at the industrial market and it offers larger networks capability, enhanced security features, reliability, and ability to change frequency channels when faced with noise and interference. Therefore a sensor node needs to have the support for the ZigBee protocol on top of the IEEE 802.15.4 protocol.

- **Integrated sensors and extensibility (Analogue/Digital ports)**

  In order to monitor the environment and the machines, sensors are required to be placed in the environment and in the machine. These sensors would need to interface and communicate with the sensor node platform hence requiring additional hardware. Therefore, it would be beneficial if the node has integrated sensors hence reducing the need for additional hardware and associated costs. If not all the required sensors are on a single sensor node then the sensor node platform needs to have the capability to allow the interfacing with external sensors. This requires that a sensor node platform have the following on-board peripherals:

  - Analogue and digital ports
  - ADC and DAC convertors with at least 12 bit resolution.
  - Counters and Timers
  - Comparators
  - Serial/UART and/or SPI/I2C Bus interface

WSN vendor	Selection Criteria					
	Processor	Memory	Radio	Wireless Protocols	Integrated Sensors	Peripherals
Crossbow (Mica2 professional kit)	ATmega 128L	4Kb RAM 128Kb flash 512Kb SRAM	2.4 GHz 868/916 MHz	IEEE 802.15.4 ZigBee	Separate board with various sensors	10bit ADC (12bit separate DAQ board)
Digi (XBee)	Freescale 16bit HCS08	32Kb flash 2Kb RAM	2.4GHz	IEEE 802.15.4 ZigBee	Separate boards	10bit ADC and DAC
Jennic (JN51xx)	32MHz RISC	192Kb ROM 96Kb RAM 256Kb flash	2.4GHz	IEEE 802.15.4 JenNET ZigBee	Temperature Humidity Light voltage	12 bit ADC 12 bit DAC
NI	533 MHz Freescale MPC834	248Kb flash	2.4GHz	IEEE 802.15.4	Various sensor nodes	16 bit ADC
TI (CC2520)	MSP430F2 274	128Kb Flash 8Kb RAM	CC2520 2.4Ghz	IEEE802.15. 4	Only I/Os for external	12 bit ADC
Telegesis	XAP16b Processor	128Kb Flash 5Kb SRAM	2.4Ghz	IEEE 802.15.4 ZigBee	Temperature Light	12 bit ADC
SUN (SunSpot)	ARM 920T (180Mhz)	8Mb Flash 1MB SRAM	2.4GHz	IEEE 802.15.4	Light Temperature Accelerometer	16 bit ADC
Meshnetics (Zigbits kit)	ATmega12 81V	8Kb RAM 128Kb Flash 4Kb EEPROM	2.4GHz	IEEE 802.15.4 ZigBee	Temperature Light	10bit ADC

Table 3.1 Selection criteria for WSN.

There are a number of WSN development kits available from various vendors with slight variations. A review of the different kits available in the market based on the above selection criteria resulted in a number of suitable choices with varying capabilities. Altogether eight kits with varying capabilities are selected as shown in Table 3.1. The kits are selected based on the on-board processing capabilities (processor and memory), transceivers, supported protocols, integrated sensors, and hardware extensibility (on-board ADC and DAC resolution). Of the eight potential kits, (Table 3.1) the processing power and memory varied. In order of processing power, the most powerful is from National Instruments (NI) which comes with a 533Mhz processor and a high-

resolution 16bit on-board ADC. Next is the Sunspot kit from Sun Microsystems which has a 180MHz processor with 8Mb of flash memory and a high-resolution 16bit ADC. At the other end is the Mica2 kit from Crossbow, which uses an 8MHz processor with a 10-bit ADC. All the selected kits use the Industrial Scientific and Medical (ISM) frequency band of 2.4GHz. The majority of them support the IEEE 802.15.4 and ZigBee protocols with the exception of the Sunspot, TI (CC2520), and NI kits. Most of the kits have either integrated temperature and light sensors or separate sensor boards at an extra cost with the exception of the TI (CC2520) kit, which has support for only external sensors through the on-board ADC.

3.2.2 **Final selection criteria**

With a wide variety of kits to choose from it was not possible to test all the selected kits due to time and budget constraints. Therefore, it was decided that further selection criteria is needed to narrow down the selection choice. The following additional criteria were chosen to narrow down the selection choice:

- Use in previous and latest research projects
- The WSN kit contents (controller, number of sensor boards, and software)
- Interactive development environment (IDE) and Programming language
- Future support and scalability
- Node power option (remote nodes with no power)
- Unit cost

The first criterion to assess the suitability of the kits for this research was to look at the use of these kits in previous research related projects. The IEEE Xplore was used to give an indication of how many times each of these kits were used in previous research. The second criterion looked at hardware and software provision with the kit. The third criterion was to look at the software development environment that was being provided with each kit. Factors such as the development programming language used, the Interactive Development Environment (IDE) support, the proprietary Application Programming Interface

WSN vendor	Selection Criteria					
	Research	Contents	Power Option	IDE	Support Scalability	Cost (£)
Crossbow (Mica2 professional kit)	>100	7 Nodes 1 Controller	Battery and External	TinyOS (nesC), Moteview Moteworks	Online/phone website and forums	900
Digi (XBee)	>100	1 Controller 4 Serial boards 1USB board	Battery and External	AT Commands (separate kit for Python SDK)	Online/Phone update from website	250
Jennic (JN51xx)	>30	1 Controller 4 Nodes	Battery and External	Complete Eclipse SDK and tool-chain	Online/Phone Regular software updates	358
NI	<10	1 Gateway 2 Nodes	Battery and External	NI LabVIEW	Online/Phone extra cost for software license	1549
TI (CC2520)	>15	2 Controllers 10 Nodes	Battery	IAR Embedded Workbench	Online/Phone extra cost	652
Telegesis	<10	1 Controller 2 Nodes	Battery	Proprietary Terminal Application	Online/Phone extra cost	249
SUN (SunSpot)	>50	1 Controller 2 nodes	Battery and External	Netbeans IDE	Open source	350
Meshnetics (Zigbit kit)	<5	1 Controller 2 Nodes	Battery and External	Atmel's AVR Studio with ZigBeeNet API	Online/Phone Extra cost after 1 year	599

Table 3.2 WSN additional selection criteria.

(API) provided with the kit and future software support. The fourth criterion is somewhat linked to the previous one in terms of the support for the software and hardware of the kit. In this criterion, factors such as current and future developments for a kit and its implications in terms of cost and support were taken into account. Although the provision of power is not affecting this research directly, (as research is being carried out in an environment where power is readily available to the kits), future work will necessitate the use of both mains and battery. Therefore, the sixth criterion was to select those kits, which provide nodes with dual power options battery and external. The final criterion was to

look at the overall cost of each. Table 3.2 shows the results using these criteria and three kits (highlighted in blue) were selected based on their use in previous research, cost, future support, power options, IDE and contents of the kit. Based on our criteria, the Jennic™ WSN; the Digi™ XBEE; and the SUN™ Microsystems SUNSPOT development kits were selected. These kits will be further tested to assess their suitability for monitoring the plastic industrial environment. This will be achieved by carrying out experiments for each kit to evaluate the following:

- The IDE and Programming environment
  - Using the IDE
    - Application development and Modification
  - The ease of configuration and setup
  - Firmware programming
- The communication capability with
  - Off-the-shelf serial programs and applications
- WSN protocols
  - IEEE 802.15.4 and ZigBee
- Sensor hardware
  - Integrated Sensors
    - Acquiring and Logging the sensor readings
  - Interface with external sensors

3.3 Conclusion

Out of the three kits evaluated, the SUNSPOT kit appeared to be the ideal choice based on the initial selection criteria as it had reasonable processing power, large on-board flash memory, RAM and a 16-bit ADC. The only shortcoming to this kit was that the transceiver only supported IEEE 802.15.4 protocol. However, during the various experiments carried out to evaluate this kit, it was found to be most unreliable. It had various problems in terms of support from the manufacturer. As this was an open source project SUN Microsystems (now Oracle) was no longer actively supporting this product. There were problems of random loss of connectivity during the running of applications and updating of the firmware on the nodes. Therefore, due to these

identified problems and the lack of future support, further testing of the SUNSPOT kit would be discontinued.

The Jennic kit came with a fully developed SDK as well as a fully developed tool-chain. There was comprehensive on-line support and documentation to compile and run some complex applications from Jennic. As well as selecting the kits for cost reasons, the Jennic JN51xx had 32 MHz on-board processor, support for 256Kb flash memory, 12bit ADC and support for three wireless protocols including the latest ZigBee protocol. As well as having the ability to support up to 500 nodes in a mesh network, it was also able to run on both; a battery or an external power source. It also comes with complete Eclipse SDK and tool chain. It however has a limited number of research papers, (only 30), using it. From the various experiments carried out to evaluate this kit, it was found to be very reliable and easy to setup with a number of scenario examples.

On the other hand, the XBee kit had comprehensive support from Digi, as it was a well-developed and supported product with a number of different versions of the radio. The other reason for selecting the XBee Series 2 was based on it also having an internal battery and the facility for external power source as well as having the ability to support up to 500 nodes in a mesh network. The other advantage of this kit was that it has been used extensively in research (>100) and could potentially be integrated with existing microcontrollers with high processing power, memory and storage. However, it did have some shortcomings such as the lack of complex examples on how to integrate the XBee devices with microcontroller hardware and in applications using Digi's Python based IDE and SDK. However, both kits have a minor drawback in that their kit contents only contain a limited number of nodes and controllers.

In the next chapter these two kits will be further tested for data reliability using the Packet Error Rate (PER) test. Based on the results from these tests ultimately one of these kits will be selected for the monitoring of the plastics machinery environment.

Chapter 4: Fidelity of the Jennic JN51xx and the XBee Series 2 based on Signal Strength and Packet Error Rate (PER).

4.1 Introduction

In this chapter, the two selected WSN development kits are tested for their communication reliability, quality of service and received signal quality and strength. To the best of the author's knowledge, there has been no study to evaluate and compare the reliability and performance of Jennic and XBee WSNs in two distinct environments. This was done using the Packet Error Rate (PER), Receive Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) parameters (see below). Furthermore, it is important to understand the benefits of using a home or an office environment, as previous studies have not made a contrast of using both. Secondly, even though some of the kits are designed for office use in mind, in this research the use of the kits for industrial purposes will be of interest. Therefore, for these reasons, the two kits are tested in two different environments to allow us to evaluate the fidelity and usage of the kits.

The PER, expressed in percentages and calculated as the ratio between the total packets sent and the number of successful packets received (by monitoring the ACK received in response) is an important Quality of Service (QoS) parameter for wireless networks. The Received Signal Strength Indicator RSSI measures the power present in an incoming signal and it is not linked to the "quality" of the signal. The RSSI value is an integer range roughly from -100 dBm to 0 dBm for IEEE 802.15.4 radios. The LQI on the other hand is a metric of the current quality of the received signal and is not related to the actual signal strength, but the signal quality often is linked to signal strength.

From literature, as mentioned, in section 2.5.2.2 it can be seen that the PER, RSSI, and LQI parameters have been used in a number of previous studies to assess the communication reliability, interferences from different sources and obstacles, and sensor node placement and localisation. These studies have included the investigation of the reliability issue in the presence of other wireless networks (Lavric et al., 2012, Simek et al., 2011) as well as in

areas perturbed by common home radio interferences and obstacles (Idoudi et al., 2013, Subaashini et al., 2013). Other studies have focused on the use of RSSI and LQI parameters for sensor node placement and localization (Halder and Kim, 2011). All these research efforts have focused on the home office and outdoor environments that are mostly composed of insulators like wood, glass, plaster, bricks, plastics etc. On the other hand, there was no study looking at industrial environments like the factory shop floor which are not considered suitable for wireless technologies. These environments are radio-hostile and most of the time consist of conductors like metals, steels, cast iron, copper etc. (Assous et al., 2009).

The XBee WSN kit is first evaluated in both the home/office and polymer industrial environments. In each of the environments, two experiments will be carried out. In the first experiment, a Low Power (LP) Coordinator is used as the transmitter and a LP End Device as the receiver. In the second experiment, a High Power (HP) Coordinator is used as the transmitter and a LP End Device is used as the receiver. Although a HP end device could have been used in these experiments it was decided to not to use this for end devices due to a few reasons including the limited number of modules available in each kit, the higher price tag associated with extra HP modules, and although energy consumption was not being investigated in this research the use of HP end devices would have a significant effect on the battery life of the sensor node.

The same experiments in the same environments will be carried out to evaluate the Jennic WSN kit. The two environments were not confined to further controlled parameters – e.g. controlling fully for background vibration and sound purely due to the physical constraints. Finally a comparison of the PER for each kit (LP and HP modules) in the two different environments is carried out and from these results one WSN is selected for monitoring a typical plastics industry environment.

4.2 Digi XBee PER Test Using X-CTU and MoltoSenso SDK

In order to carry out the range, packet transmit, and receive tests; a software is required which can at least transmit and receive the packets and has the capability to check if a packet has arrived without errors. A survey of the available packet sniffing software in the market showed that majority of these

were proprietary in nature. These packet sniffers require their own dedicated additional hardware in order to monitor a WSN in the vicinity, which adds cost and complexity to the experiments. There was no software as such that would support all ZigBee devices regardless of their manufacturers. It was therefore decided that the X-CTU (Digi International, 2010b) configuration and test utility software from Digi will be used in conjunction with Moltosenso Network Manager™ (MNM) (Urso, 2012) a powerful SDK to setup and manage Digi International modules for this purpose. Both software had the ability to monitor the number of packets sent and received as well as showing the PER and RSSI values.

4.2.1 Experiment 1: PER and RSSI Test using XBee Series 2 Low and High Power Modules in Home/Office Environment.

This experiment was carried out in two parts. In the first part of this experiment, two XBee Series 2 LP radios with an indoor range of up to 30 meters were used to verify the data packet delivery from the transmitter to receiver. The modules were then used to carry out the PER and RSSI tests. In the second part of this experiment, one XBee Pro Series 2 HP module with an indoor range of up to 90 meters was used as the Coordinator node whilst a LP module was used as the Router/End Device to carry out the PER and RSSI tests. The full specifications of the LP and HP XBee modules can be found on the Digi website (Digi International, 2010c). The aim of this experiment was to:

- Verify the data packets being sent and received.
- Analyse and compare the PER at various separation distances between LP and HP modules.
- Analyse and compare the RSSI at various separation distances between LP and HP modules

4.2.1.1 *Experimental Setup*

Environment: Lynwood Home/Office environment

Hardware: 1 XBee Pro Series2 radio (RPSMA antenna), 1 XBee Series 2 radio (wire antenna), 1 XBee Series 2 radio (chip antenna), 1 RS232 board, 1 USB board and 1 serial loopback plug.

Firmware: XBee Series 2 ZNet 2.5

- The XBee Pro Series 2 HP radio (RPSMA antenna) was programmed as Coordinator
- The XBee Series 2 LP radio (wire antenna) was programmed as Coordinator
- The XBee Series 2 LP radio (chip antenna) was programmed as an Router/End Device

Software:

- X-CTU terminal software
- MoltoSenso Network Management

Protocol: XBee ZNet 2.5 (ZigBee 2006 standard)

Connection: Serial/USB

Part one - PER and RSSI Test using two LP Modules: The PER and RSSI data was collected at various separation distances between the two LP nodes. This was achieved by attaching the serial loopback plug to the Router/End Device node so that it will send the received data packets back to the transmitter. The MNM software described in more details in (Urso, 2012) has the ability to connect to the XBee Coordinator node in a WSN through the COM port of the PC. Once the Coordinator is connected the network discovery command can be used to detect all the nodes in the WSN including the Coordinator connected via the COM port. After the network discovery command, if remote nodes have been found, then it is possible to perform an RSSI test between the current device connected via the COM port and one of the discovered remote devices.

Figure 4.1 shows the MNM RSSI test Tab, the first two widgets in the middle of the Tab show the RSSI value measured on both the local node (on the left) and on the selected remote node (in the middle). They report the instantaneous numerical value and the corresponding representation with a bar graph. The widget on the right provides the number of sent and received packets and the PER which is the ratio between sent and correctly received packets. In this experiment, the RSSI test was carried out in the Lynwood

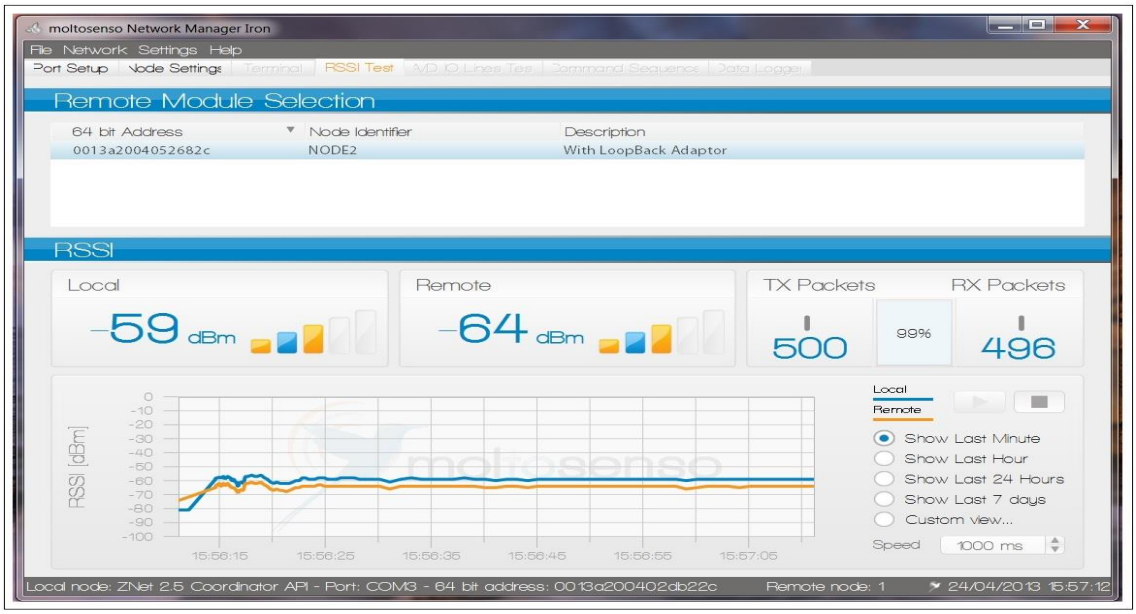


Figure 4.1 The MNM RSSI test Tab.

home/office environment at three separation distances between the two nodes as shown in Figure 4.2. At a separation distance of 5 meters between the Coordinator and the Router/End Device, the RSSI test was carried out by sending 500 data packets at intervals of 300ms. This test was repeated three times to get an average PER and RSSI value at this separation distance. The same procedure was repeated at 10 meters and 15 meters separation distances.

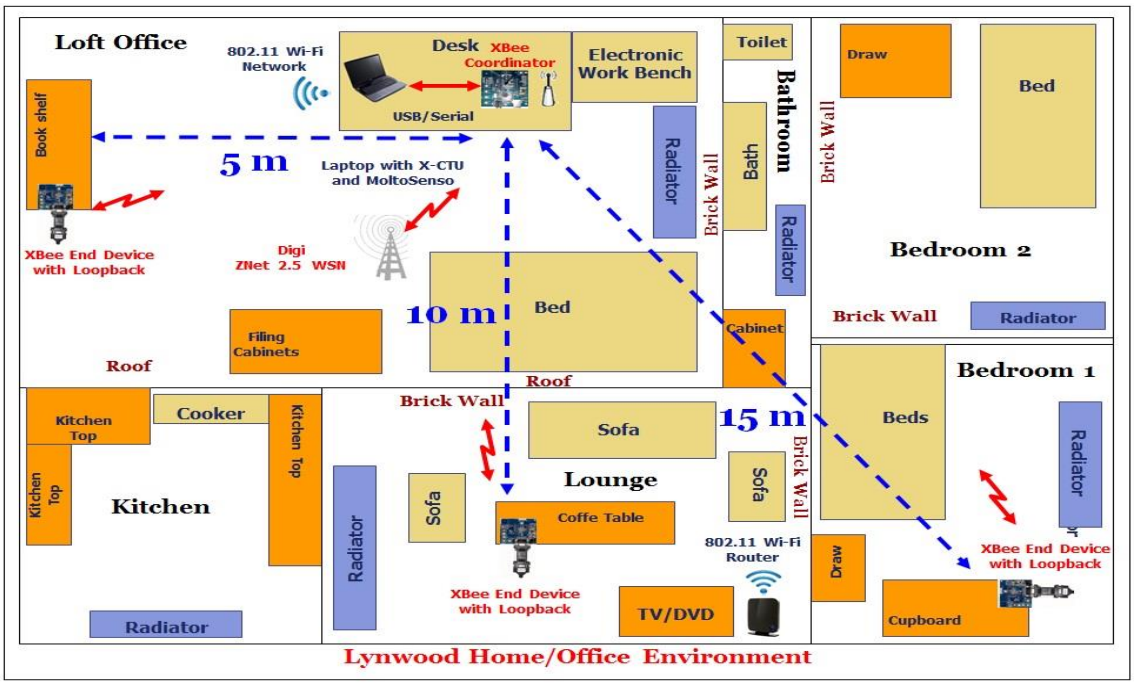


Figure 4.2 PER and RSSI test at three separation distances in Lynwood Home/Office Environment.

Part two - PER and RSSI test using a HP Coordinator Module: This part of the experiment is almost identical to part one of this experiment with the only difference being the LP Coordinator being swapped with the XBee Pro series 2 HP module. The same RSSI and PER tests were carried out in the Lynwood home/office environment at three separation distances between the two nodes as shown in Figure 4.2.

4.2.1.2 *Results*

Data Verification: The data verification was done by sending messages from the MNM software to the X-CTU utility using the two XBee S2 nodes. The X-CTU utility successfully received and displayed the messages in its terminal window every three seconds as shown in Figure 4.3.

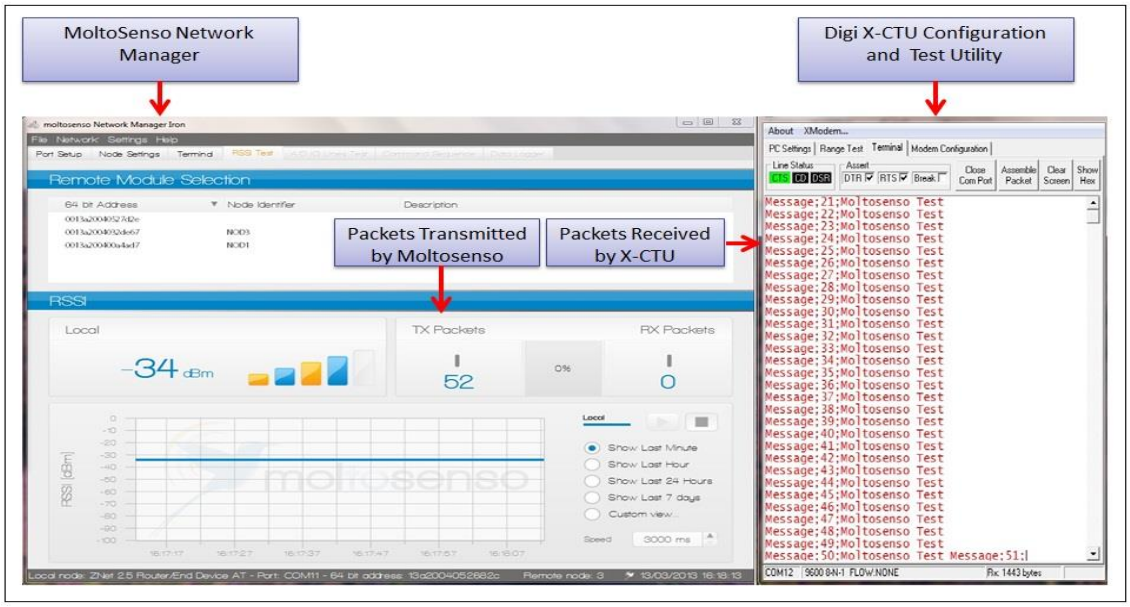


Figure 4.3 Test message verification.

Part One and Two - RSSI and PER using LP, HP Modules in Home/Office Environment:

The averaged RSSI and PER data was analysed at three separation distances (5m, 10m, 15m) for both the LP modules and a HP Coordinator module. It could be observed from the results that when LP modules are used, as the distance increases the RSS gets weaker. However, when a HP module was used as a Coordinator, the RSS improved significantly at closer range. The signal loss is

significantly less when using a HP Coordinator, which is as expected with a HP module with an indoor range of up to 90 meters.

It can also be noted that as the separation distance is increased the PER starts to increase. At 5 meters separation distance the PER for both the LP and HP module is virtually zero that is a very reliable communication link. But as the distance is increased to 10 meters the PER increases as well but not significantly as it stays between 0.03% and 0.05 % for both the LP and HP modules. However, it is clear at this range that the LP modules exhibit a slightly higher PER. At a separation distance of 15 meters the PER for LP modules is 0.2 %, an increase of almost three times compared to the 10 meters PER. When using the HP Coordinator the PER is significantly less compared to the LP modules at this range. Overall for both the LP and HP modules the PER is almost zero (average of 0.06%) and the communication can be considered reliable at all three separation distances when compared to the conclusion of (Subaashini et al., 2013) which states that a PER of less than 2% can be considered as a reliable communication. It can also be concluded that when a HP Coordinator is used the communication reliability increases significantly at longer distances.

4.2.2 Experiment 2: PER and RSSI Test using XBee Series 2 Low and High Power Modules in Polymer Industrial Environment.

This experiment is fundamentally very similar experiment to the one in section 4.2.1 with a few exceptions. It is not necessary to verify data transmission here, as the functionality of the system has already been determined. The same XBee modules and the same set of procedures is followed as in parts one and two of experiment one with the only difference being that these tests are carried out in the polymer industrial environment shown in Figure 4.4.

The aim of this experiment was to:

- Analyse and compare the RSSI at various separation distances between LP and HP modules in an industrial environment.
- Analyse and compare the PER at various separation distances between LP and HP modules in an Industrial environment.

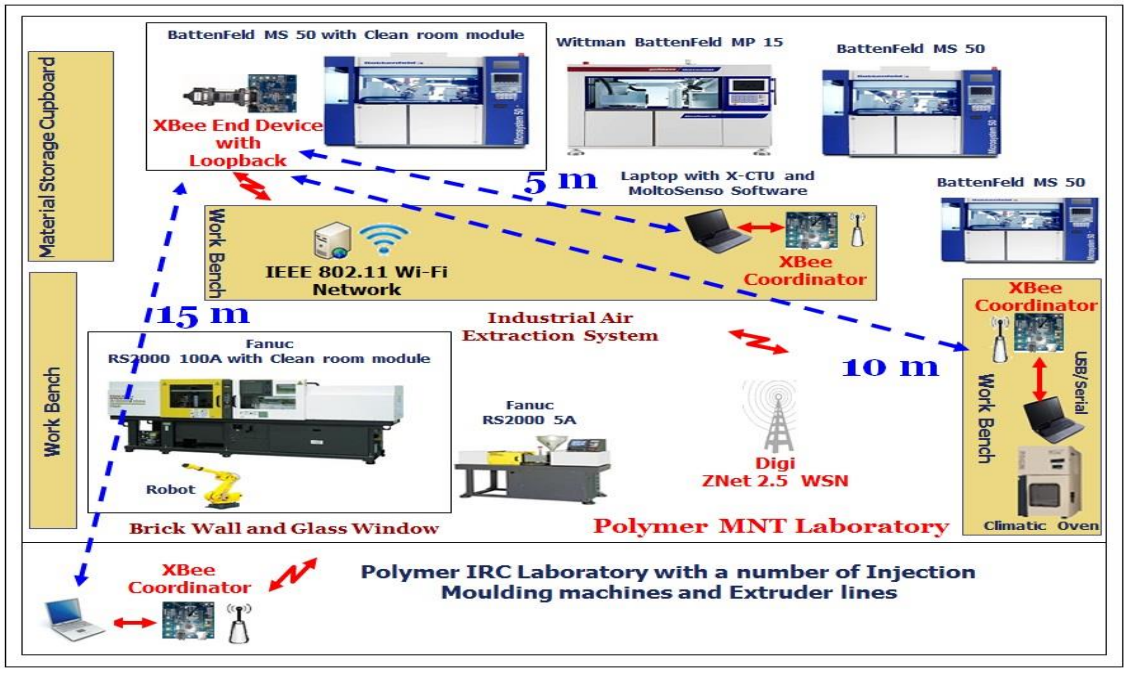


Figure 4.4 PER and RSSI test at three separation distances in Polymer Industrial Environment.

4.2.2.1 *Experimental Setup*

Environment: Polymer IRC Laboratory Industrial Environment

Hardware: 1 XBee Pro Series2 radio (RPSMA antenna), 1 XBee Series 2 radio (wire antenna), 1 XBee Series 2 radio (chip antenna), 1 RS232 board, 1 USB board and 1 serial loopback plug.

Firmware: XBee Series 2 ZNet 2.5

- The XBee Pro Series 2 HP radio (RPSMA antenna) was programmed as Coordinator
- The XBee Series 2 LP radio (wire antenna) was programmed as Coordinator
- The XBee Series 2 LP radio (chip antenna) was programmed as an Router/End Device

Software:

- X-CTU terminal software
- MoltoSenso Network Management software

Protocol: XBee ZNet 2.5 (ZigBee 2006 standard) fun

Connection: Serial/USB

Part one - PER and RSSI Test using two LP Modules: In this experiment the RSSI test was carried out in the Polymer IRC industrial environment at three separation distances as shown in Figure 4.4. The Coordinator node was connected to the Laptop with the MNM software shown in Figure 4.5. The End Device node had a serial Loopback adaptor connected to it and it was installed in a Battenfeld MS50 µIM machine shown in Figure 4.6.

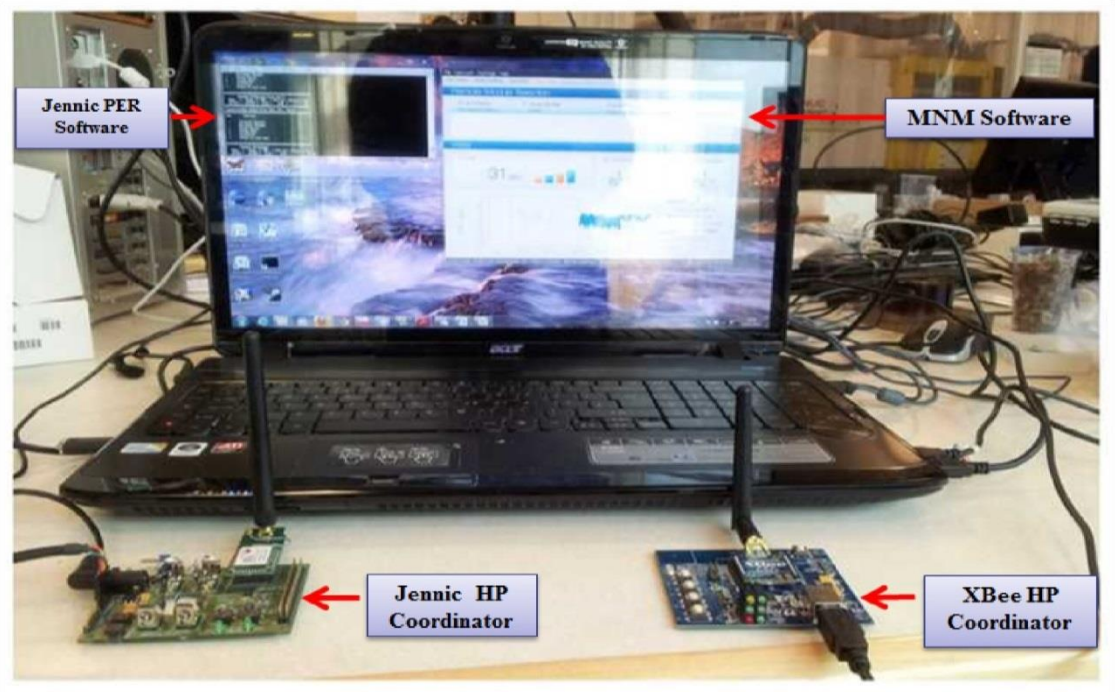


Figure 4.5 XBee and Jennic Coordinators connected to a Laptop

whilst it was running a manufacturing job. At a separation distance of 5 meters between the Coordinator and the Router/End Device, the RSSI test was carried out by sending 500 data packets at intervals of 300ms. This test was repeated three times to get an average PER and RSSI value at this separation distance. The same procedure was repeated at 10 meters and 15 meters separation distances.

Part two - PER and RSSI test using a HP Coordinator Module: This part of the experiment is almost identical to part one of this experiment with the only difference being the LP Coordinator being swapped with the XBee Pro series 2 HP module. The same RSSI and PER tests were carried out in the Polymer IRC industrial environment at three separation distances between the two nodes as shown in Figure 4.4.

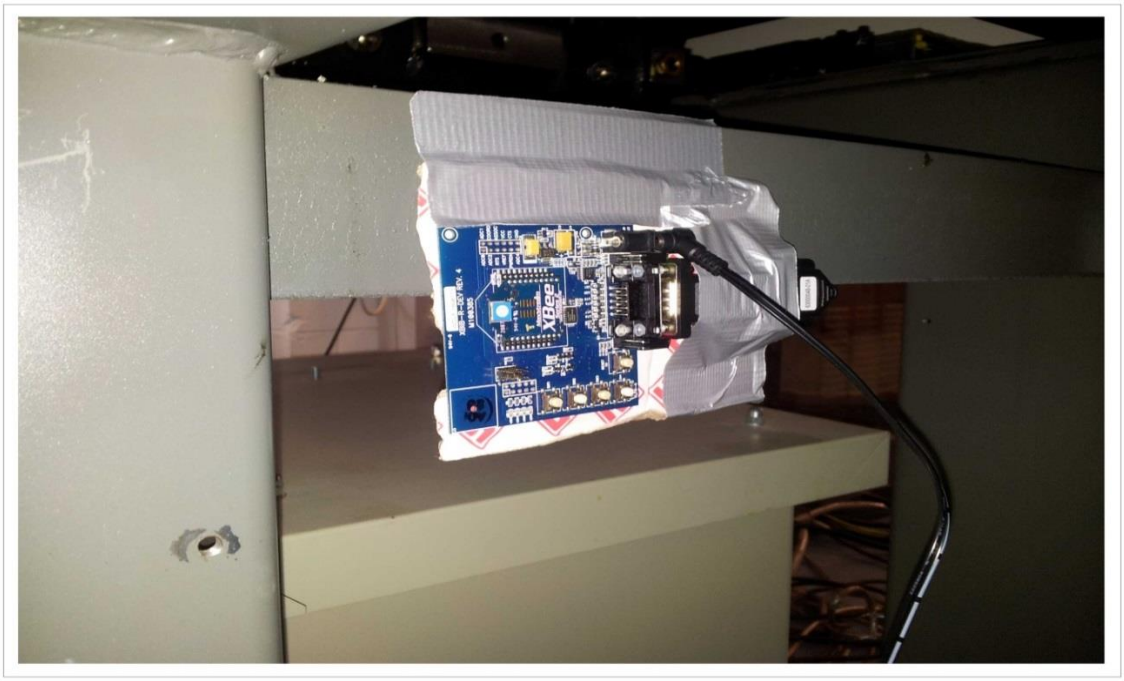


Figure 4.6 XBee Router/End Device with Loopback Adaptor installed in a µIM machine.

4.2.2.2 . *Results*

Part one and two - Comparison of RSSI and PER using LP and HP Modules in Polymer Industrial Environment:

The averaged RSSI and PER data was analysed at three separation distances (5m, 10m, 15m) for both the LP modules and a HP Coordinator module. From the results, a similar trend could be observed to that of the home/office environment. When all LP modules are used in this test, the highest RSS was around -56 dBm at 5 meters separation distance. The lowest RSS was around -75 dBm at 15 meters separation distance. So as the distance increases the RSS gets weaker. When a HP module is used as a Coordinator, the RSS improved significantly at closer range. The highest RSS was -47 dBm and the lowest RSS was -59 dBm. The signal loss is significantly less when using a HP Coordinator, which is as expected with an indoor range of up to 90 meters.

It can also be noted that as the separation distance is increased the PER starts to increase. At 5 meters separation distance the PER for both the LP and HP module is zero which means that a very reliable communication link is present at this range. But as the distance is increased to 10 meters the PER for all LP modules is 0.1 % and it is double to that of when a HP Coordinator is used which is 0.05%. At a separation distance of 15 meters the PER for LP

modules is 0.4 %, an increase of almost 3 times compared to the 10 meters PER range. When using the HP Coordinator the PER is again almost twice as less compared to the LP modules at this range. Overall for both the LP and HP modules the PER is between 0.05% and 0.4% which can be considered reliable at all three separation distances when compared to the conclusion of (Subaashini et al., 2013) which states that a PER of less than 2% can be considered as a reliable communication. It can also be concluded that when a HP Coordinator is used the communication reliability increases significantly at longer distances.

4.2.3 Comparison of Home/Office and Polymer Industrial Environments

In this section a comparison of the results for the XBee modules in the home/office and polymer industrial environments is done. First a comparison of the RSSI parameter in both environments is carried out. Then a comparison of the PER parameter is done.

4.2.3.1 *Comparison of RSSI in Home/Office and Polymer Industrial Environment*

This part of the results shows a comparison done for the RSSI parameter, which has been measured in both the home/office and polymer industrial environments. First a comparison to evaluate the performance of all LP modules in both environments is done. Figure 4.7 shows the results of this comparison. It can be seen from the left hand side graph that when using all LP modules at 5 meters separation distance the RSSI is almost identical for both the industrial and home/office environments as there are no major obstacles such as walls in both the environments. However, the industrial environment does experience slightly higher signal loss at 10m and above distance. This can be attributed to the large equipment in the environment such as the µIM machines and the dynamic nature of the environment such as moving parts in the machinery and people walking between the transmitter and receiver. A comparison to evaluate the performance of a HP Coordinator module in both environments was done and the graph on the right in Figure 4.7 shows the results of this comparison. When a HP module was used as the transmitter, at 5 meters separation distance the RSSI results are very similar to the LP modules as there are no major obstacles such as walls in both the environments. For

both the LP and HP modules, at longer distances, in the home/office environment the RSSI indicates higher signal loss compared to the same industrial environment. This can most likely be attributed to the fact that there were more than one brick wall obstacles between the transmitter and the receiver in the home/office environment. Overall, it can be seen that with the use of a HP module as the transmitter the RSS is significantly better.

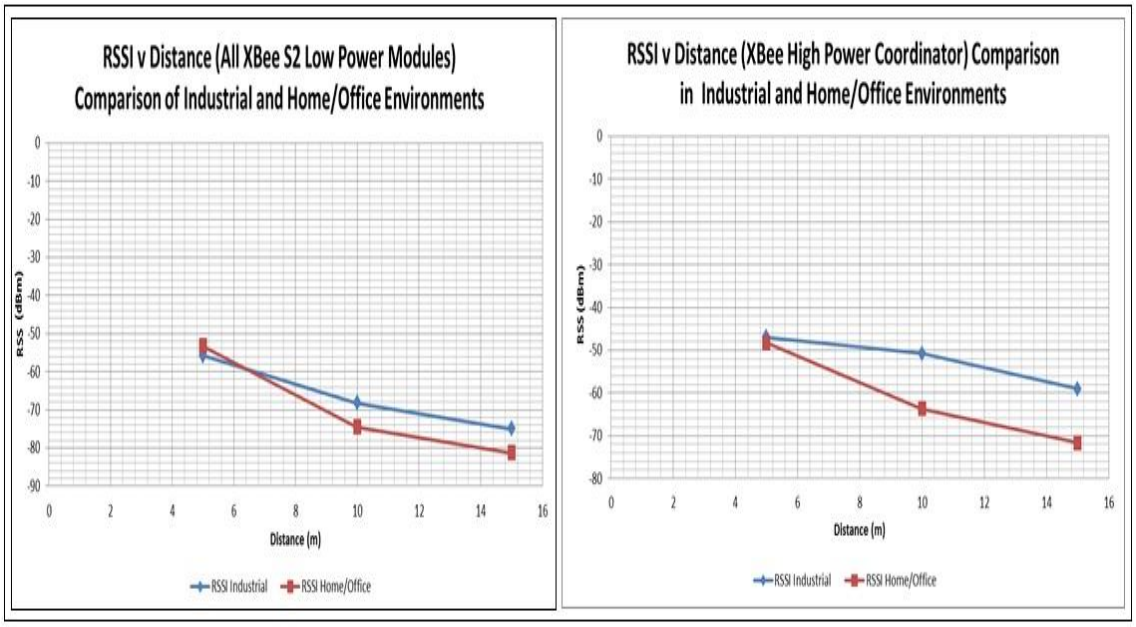


Figure 4.7 RSSI comparison in home/office and polymer industrial environments

4.2.3.2 *Comparison of PER in Home/Office and Polymer Industrial Environment*

This part of the results shows a comparison done for the PER parameter which has been measured in both the home/office and polymer industrial environments. First a comparison to evaluate the performance of all LP modules in both environments is done and the results are shown in the left hand side graph in Figure 4.8. At 5 meters separation distance when using all LP modules the PER for both the home/office and polymer industrial environments is zero indicating a very reliable communication link. But as the distance is increased to 10 meters the PER for all LP modules in an industrial environment is almost double to that of the home/office environment and this trend continues for the 15 meter range. As the distance increases the PER increases in both environments although the industrial environment exhibits almost two times as

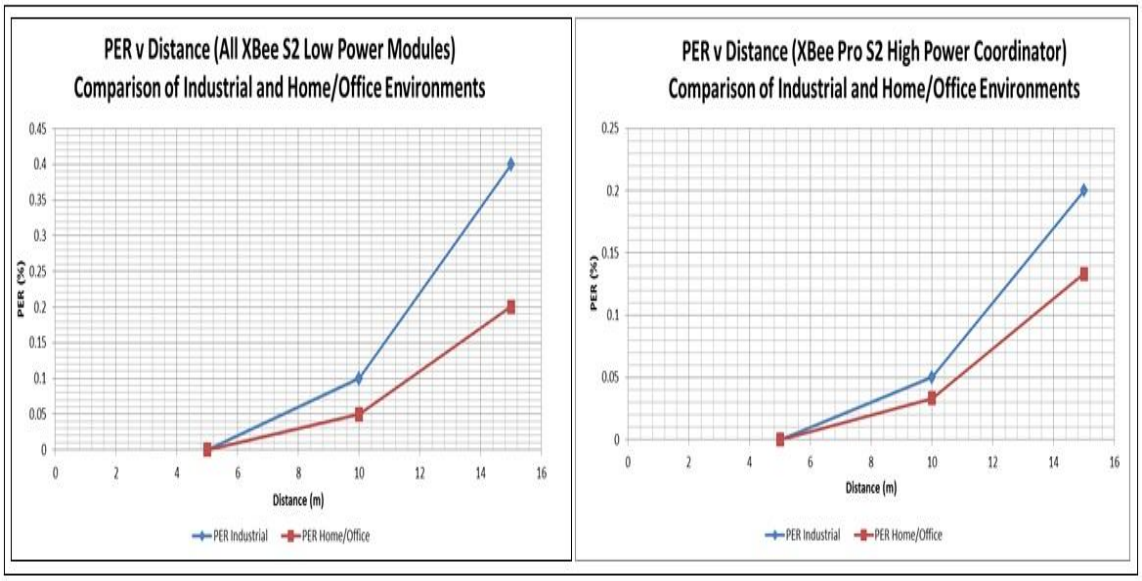


Figure 4.8 Comparison of PER using all LP and HP modules in home/office and polymer industrial environments.

much PER at distance of 10m and above. The graph on the right hand side in Figure 4.8 shows the results of using a HP Coordinator module in both environments. At 5 meters separation distance the PER for both the home/office and polymer industrial environments is zero indicating a very reliable communication link. As the distance increases to 10 meters and above the PER in an industrial environment is 50% more compared to the home/office environment. In addition, to note is that when using a HP module as the transmitter, the industrial environment exhibits fewer errors compared to the home/office environment.

Overall it can be concluded that the PER is higher in the industrial environment compared to the home/office environment. This can be attributed to a number of factors including the high temperatures, moving parts, people moving, brick walls, and the Wi-Fi network in the same vicinity. Using the XBee modules the PER is still between 0 and 0.4% and this can be considered as a reliable communication link.

4.3 Jennic JN5148 PER Test using the JN-AN-1006 PER test Software

The PER test for the Jennic WSN kit was done by using the JN-AN-1006 PER test software from Jennic (Jennic Ltd, 2011). This software allows PER testing to be conducted using boards from the Jennic JN5148, JN5142 or JN5139

evaluation kits. Two application binary files are provided, one for a PER Master and another for a PER Slave. The PER Master and Slave binary files are uploaded onto nodes connected to a PC via serial/USB port. Packets (frames) are sent between the boards and the PER results are displayed on a serially connected PC using the TeraTerm terminal software. This test does not require a Loopback adaptor as the software can automatically handle the replies to each data packet. The software sends a test data packet every 20ms using the transmitter node and the receiver sends an acknowledgement back if the data packet is received successfully. With each packet, the local LQI is also shown. Using the Jennic JN51xx modules the RSSI can be linearly related to the LQI using the following equation (Yao et al., 2008).

$$i16RSSI = \left(u8LQI \times \left(\frac{880000000}{255}\right)/10000000\right) - 98$$

Equation 4.1 LQI to RSSI conversion. This equation gives a linear relationship between the LQI and the RSSI.

Where, i16 and u8 respectively denote 16-bit integer and 8-bit unsigned integer in the C programming language. The RSSI ranges from −98 dBm to −10 dBm, which covers through ZigBee minimum to maximum sensitivity. In practice, the LQI is a rough indicator and therefore the associated received signal power measure has large variance. The RSSI was calculated using Equation 4.1 for each test using the LQI value recorded during the test.

4.3.1 Experiment 1: PER and RSSI Test using Jennic 5148 M03 LP and M04 HP Modules in Lynwood Home/Office Environment.

This experiment was carried out in two parts. In the first part of this experiment two Jennic M03 LP radios with an indoor range of up to 50 meters were used to carry out the PER and RSSI tests. In the second part of this experiment, one Jennic M04 HP module with an indoor range of up to 200 meters was used as the Coordinator node whilst a LP module was used as the End Device to carry out the PER and RSSI tests. The full specifications of the LP and HP Jennic modules can be found on the Digi website (Digi International, 2010c). The aim of this experiment was to:

- Analyse and compare the RSSI at various separation distances between LP and HP modules in the Lynwood home/office environment

- Analyse and compare the PER at various separation distances between LP and HP modules in the Lynwood home/office environment.
- Analyse the effects of varying the packet re-transmission (retries) on the PER in the Lynwood home/office environment.

4.3.1.1 *Experimental Setup*

Environment: Lynwood Home/Office environment

Hardware: 2 Jennic JN5148-001-M03 (RPSMA antenna) LP modules, 1 Jennic JN5148-001-M04 (RPSMA antenna) HP module, 2 DR1048 sensor boards.

Firmware: JN-AN-1006

- The Jennic M04 HP radio (RPSMA antenna) was programmed as the Master.
- One Jennic M03 LP radio (RPSMA antenna) was programmed as the Master
- One Jennic M03 LP radio (RPSMA antenna) was programmed as the Slave

Software:

- JN-AN-1006 PER test application

Protocol: JenNet (IEEE 802.15.4)

Connection: Serial/USB

Part one - PER and RSSI Test using two LP Modules: In this experiment the JN-AN-1006 PER test application software (Jennic Ltd, 2011) was used to carry out the PER test. The TeraTerm terminal utility was used to run the PER test application residing on the Master Coordinator via the serial COM port on a PC. This application allows the selection of the relevant master module (LP or HP) for the test. Once the relevant master module has been selected, an option of running a number of tests comes up from which the connectionless PER test was selected. The software then gives the option of selecting the number of retries, frequency channel, and the test mode (with or without acknowledgements). For the number of retries the IEEE 802.15.4

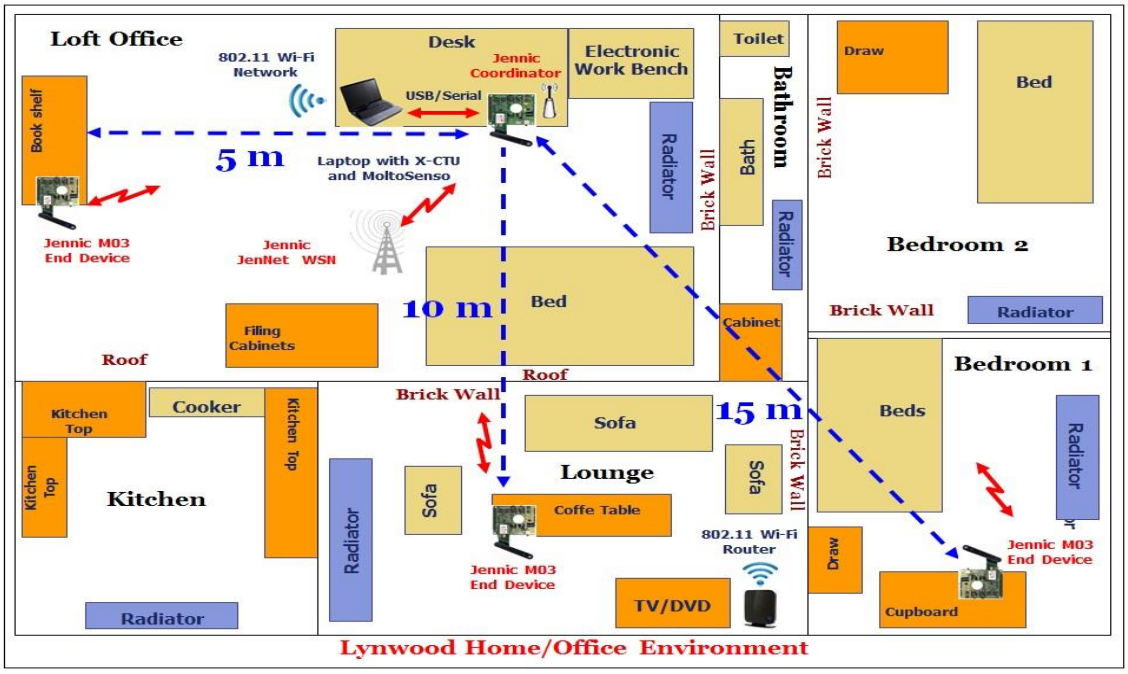


Figure 4.9 Jennic PER test at three separation distances in home/Office environment.

.

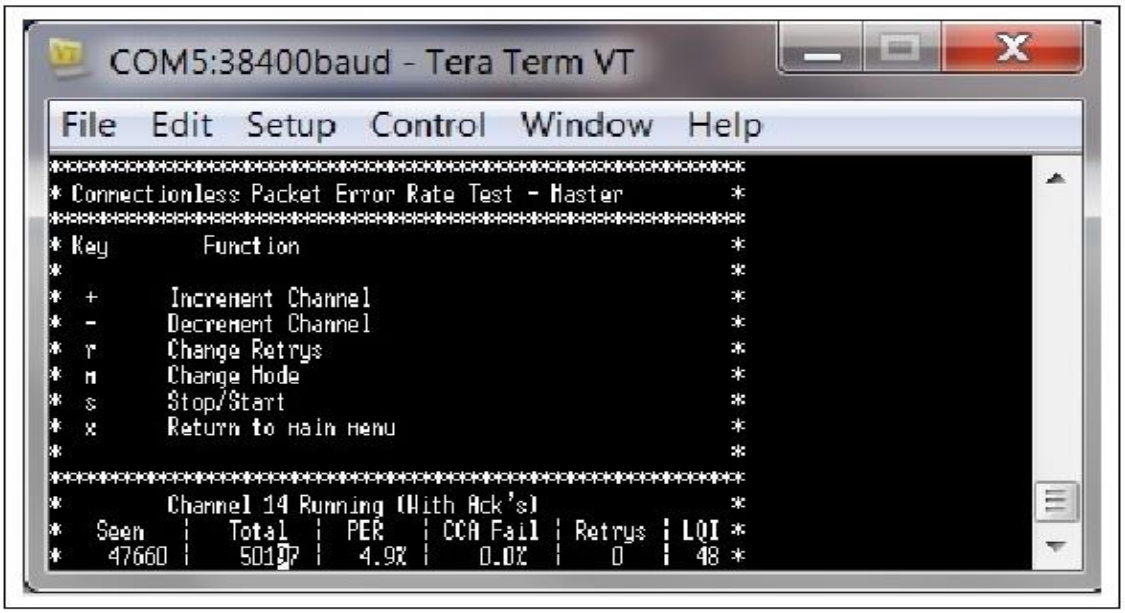


Figure 4.10 The JN-AN-1006 PER test running in the TeraTerm utility.

standard suggests a minimum of three retries if a packet is not received correctly for optimal results. In this experiment the PER test was carried out in the Lynwood home/office environment at three separation distances (5m, 10, 15m) between the two nodes as shown in Figure 4.9. In real-time data communications, message retries are not considered good for the latency of message delivery as they introduces indeterminism and this can lead to unacceptable delays up to a minute. However, it does have advantages when it

comes to reliability. In this project, the real-time response of the system is not as important as the reliability since the main aim is to monitor the key parameters of the process and not control them. The control and real-time response of the system will be considered as part of the future work. Therefore for each separation distance the number of retries was also varied (0, 3, and 7) and the test was repeated three times to get an average result for the PER and LQI. In total for each separation distance, 9 tests were carried out and for each test, 50000 data packets were sent from the Master Coordinator to the Slave End Device at intervals of 20ms. The master node transmits data packets to the slave node on a frequency channel (11 to 26) automatically selected by the software based on the best-received signal quality. If the packets were received correctly then an acknowledgement is sent back to the master to confirm this. The master keeps a count of the acknowledgements received. The results are displayed as percentages of successful packets received (100% corresponds to all packets received) as shown in Figure 4.10.

Part two - PER and RSSI Test using a HP Coordinator Module: This part of the experiment is almost identical to part one with the only difference being the LP Jennic M03 Coordinator being swapped with the Jennic M04 HP module. The PER tests were carried out in the Lynwood home/office environment at three separation distances between the two nodes as shown in Figure 4.9.

4.3.1.2 *Results*

Part one and two - Comparison of the RSSI and PER using LP and HP Modules in Home/Office Environment:

For each separation distance, three tests were carried out and the average was calculated based on the three results. The averaged LQI data (for the standard three retries) was converted into RSSI value and this was plotted against the three separation distances for both the LP modules and a HP Coordinator module. Table 4.1 shows the RSSI comparison results for LP only modules and HP Coordinator module. It can be seen from this result that when all LP modules are used in this test the highest RSS was around -62 dBm at 5 meters

RSSI using Jennic modules in Home/Office Environment		
Distance (meters)	**RSS (dBm) LP**	**RSS (dBm) HP**
5m	-62	-43
10m	-78	-64
15m	-92	-74

Table 4.1 Jennic RSSI when using all LP and a HP Coordinator modules.

separation distance. The lowest RSS was around -92 dBm at 15 meters separation distance. So as the distance increases the RSS gets weaker. But when a HP module is used as a Coordinator the RSS improved by an average of almost 22% which is a significant increase and is as expected with a HP module with an indoor range of up to 200 meters.

The averaged PER using standard three retries was plotted against the three separation distances for both the LP modules and a HP Coordinator module. Table 4.2 shows the PER comparison result for LP only modules and HP Coordinator module. From this result it can be noted that as the separation distance is increased the PER starts to increase. At 5 meters separation distance the PER for both the LP and HP module is virtually zero that is a very reliable communication link. But as distance is increased to 10 meters the PER increases as well but not significantly as it stays between 0.012% and 0.011 % for both the LP and HP modules. At a separation distance of 15 meters the PER for LP modules is 0.221 %, a significant increase when compared to the 10 meters PER. This can most likely be attributed to the dynamic nature of the environment and other interferences such as satellite or microwave signals.

PER using Jennic modules in Home/Office Environment		
Distance (meters)	**PER (%) LP**	**PER (%) HP**
5m	0.003	0
10m	0.012	0.011
15m	0.221	0.004

Table 4.2 Jennic PER when using all LP and a HP Coordinator modules.

When using the HP Coordinator at 15m distance the PER does not change and is slightly better and compared to the LP module the PER is significantly less. Overall for both the LP and HP modules the PER is between 0 and 0.221% that is almost zero and the communication can be considered reliable at all three separation distances when compared to the conclusion of

(Subaashini et al., 2013) which states that a PER of less than 2% can be considered as a reliable communication. It can also be concluded that when a HP Coordinator is used the communication reliability increases significantly at longer distances.

Part one and two - Analysis of the effects of varying the packet re-transmission (retries) on the PER using LP and HP Modules in Home/Office Environment:

For each separation distance the PER test was carried out for three different retry values (0, 3 and 7). The aim was to see how this would impact on the PER using all LP modules and a HP Coordinator module. The test was repeated three times to get an average value for the PER. The graph on the left in Figure 4.11 shows the PER for different retries at three separation distances using all Jennic M03 LP modules. From this result it can be seen that when the number of retries is kept to zero the PER jumps significantly from almost zero to a maximum of 1.215% at 5 meters separation distance. At 10m distance the PER was expected to go up as the distance increases but the result shows an anomaly with a significant drop in the PER to almost 0.8%. It goes back up to 1.2% at 15m distance. This unexpected result could most likely be attributed to the interference of a satellite broadband signal or a microwave signal in the near vicinity. The signal must have been initially present hence giving rise to a higher PER at short distances. At 10m distance, the interference signal is most likely absent resulting in a lower PER. When the number of retries is increased to the maximum seven allowed in this software the PER remains zero at all three separation distances. This is an excellent result in terms of communication reliability but this is only possible as the data packet size is small (32 bytes). This size is adequate for this project, but if this was to be increased say to 64 bytes then a rise in PER will be seen due to the time required to successfully process the data packets in the queue and sending and receiving acknowledgements. The graph on the right hand side in Figure 4.11 shows the results for the different retry values when using a HP Coordinator as the master transmitter. The results show almost a PER of zero for three and seven retries for all three separation distances. However, this is not the case when the number of retries is set to zero and the PER jumps significantly to a

maximum of 1.870% at 10m distance. Although the PER was expected to increase at 10m distance, the significant increase can be attributed to the presence of a satellite broadband signal or a microwave signal in the near vicinity. In a home/office environment it can be concluded that using the standard three retries as recommended by the IEEE 802.15.4 standard will give optimal results in terms of PER.

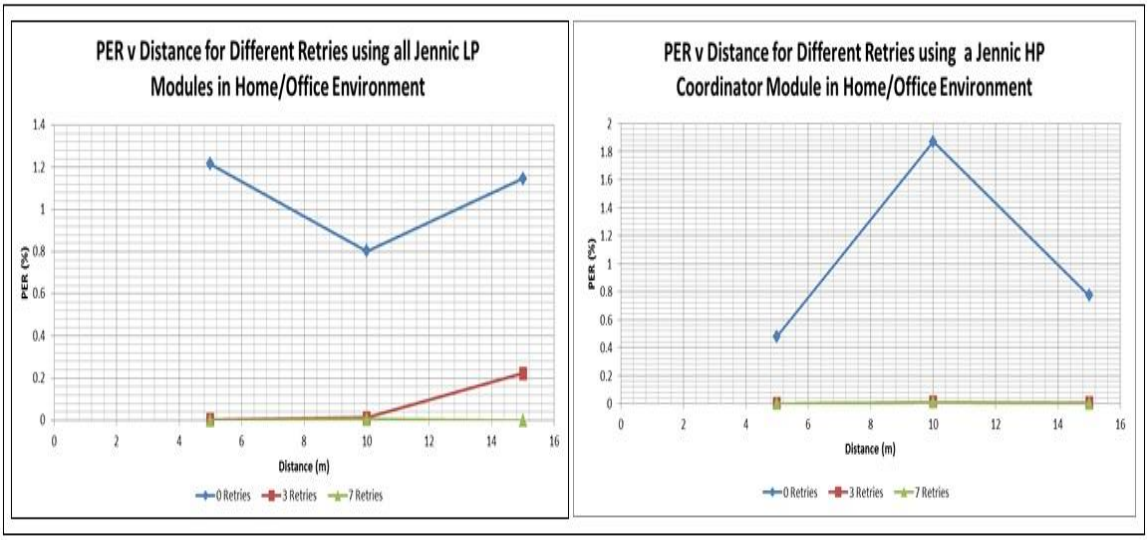


Figure 4.11 PER for different retries in home/office environment using Jennic M03 LP modules.

4.3.2 Experiment 2: PER and RSSI Test using Jennic 5148 M03 LP and M04 HP Modules in Polymer Industrial Environment.

This experiment had two parts to it and is the same as experiment one in section 4.3.1 with a few exceptions. The same Jennic modules and the same set of procedures is followed as in parts one and two of experiment one with the only difference being that these tests are carried out in the Polymer industrial environment.

The aim of this experiment was to:

- Analyse and compare the RSSI at various separation distances between LP and HP modules in the Polymer industrial environment.
- Analyse and compare the PER at various separation distances between LP and HP modules in the Polymer industrial environment.
- Analyse the effects of varying the packet re-transmission (retries) on the PER in the Polymer industrial environment.

Experimental Setup

Environment: Polymer industrial environment

Hardware: 2 Jennic JN5148-001-M03 (RPSMA antenna) LP modules, 1 Jennic JN5148-001-M04 (RPSMA antenna) HP module, 2 DR1048 sensor boards.

Firmware: JN-AN-1006

- The Jennic M04 HP radio (RPSMA antenna) was programmed as the Master.
- One Jennic M03 LP radio (RPSMA antenna) was programmed as the Master
- One Jennic M03 LP radio (RPSMA antenna) was programmed as the Slave

Software:

- JN-AN-1006 PER test application

Protocol: JenNet (IEEE 802.15.4)

Connection: Serial/USB

Part one and two - PER and RSSI Test using two LP and a HP Module: In this experiment the RSSI and PER tests were carried out in the Polymer IRC laboratory. The Jennic Coordinator node was connected to the Laptop with the

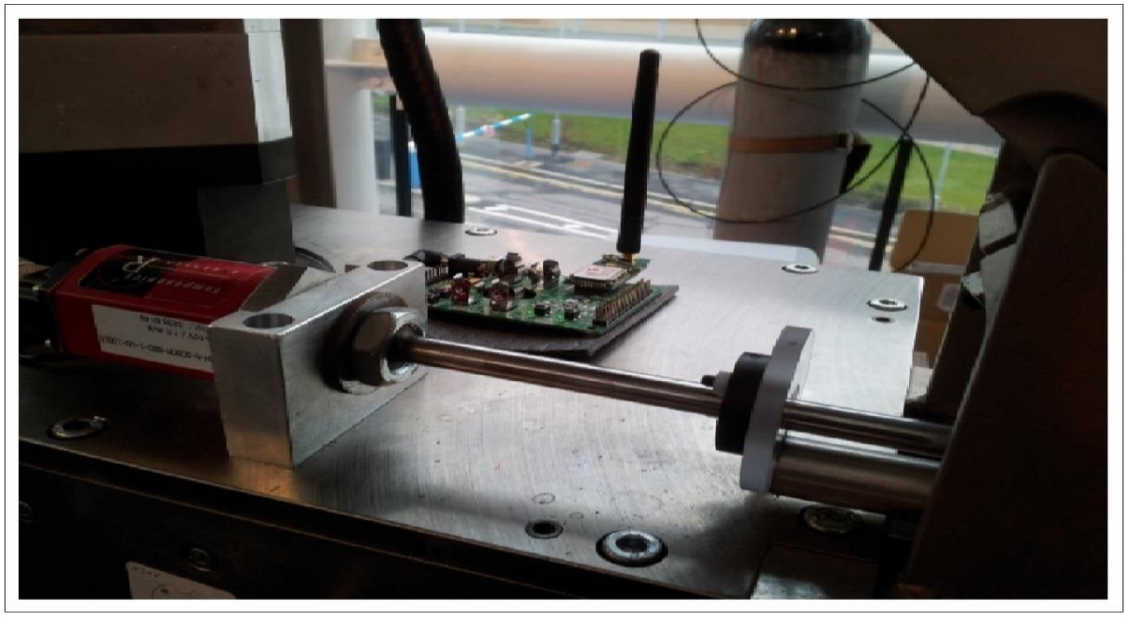


Figure 4.12 Jennic Slave End Device installed in a µIM machine.

JN-AN-1006 PER test application software running in a terminal window as shown in Figure 4.5. The Jennic End Device node was installed in a Battenfeld MS50 µIM machine shown in Figure 4.12 whilst the machine was running a manufacturing job. The test was carried out at each separation distance by sending 50000 data packets from the Master Coordinator to the Slave End Device at intervals of 20ms. For each separation distance, the test was repeated by varying the number of retries (0, 3, and 7). This procedure was repeated three times to get an average result for the PER and RSSI. Therefore, a total of nine tests were carried out for each separation distance.

4.3.2.1 *Results*

Part one and two - Analysis of the RSSI and PER using LP and HP Modules in Polymer Industrial Environment:

For each separation distance, three tests were carried out and the average was calculated based on the three results. The averaged LQI data (for the standard three retries) was converted into RSSI value and this was plotted against the three separation distances for both the LP modules and a HP Coordinator module. Table 4.3 shows the RSSI comparison results for LP only modules and HP Coordinator module. It can be seen from this result that when all LP modules are used in this test the highest RSS was around -71 dBm at 5 meters separation distance.

RSSI using Jennic modules in Polymer Industrial Environment		
Distance (meters)	**RSSI (dBm) LP**	**RSSI (dBm) HP**
5m	-71	-67
10m	-75	-73
15m	-80	-77

Table 4.3 Jennic RSSI when using all LP and a HP Coordinator module in the polymer industrial environment.

The lowest RSS was around -80 dBm at 15 meters separation distance. So as the distance increases the RSS gets weaker. However, when a HP module is used as a Coordinator the RSS improved by an average of 4%. The signal loss is slightly less when using a HP Coordinator which is as expected with a HP module with an indoor range of up to 200 meters. The averaged PER using standard three retries was recorded against the three separation distances for

PER using Jennic modules in Polymer Industrial Environment		
Distance (meters)	PER (%) LP	PER (%) HP
5m	0.078	0.036
10m	0.299	0.175
15m	0.475	0.227

Table 4.4 Jennic PER when using all LP and a HP Coordinator modules in polymer industrial environment.

both the LP modules and a HP Coordinator module. Table 4.4 shows the PER comparison result for LP only modules and HP Coordinator module. From this result it can be noted that as the separation distance is increased the PER starts to increase. At 5 meters separation distance the PER for the LP only modules is 0.078% but for the same distance when using a HP Coordinator module the PER is reduced by over 50%. As we increase the distance to 10 meters the PER increases for both the LP only and HP Coordinator modules. When using the HP module at this distance a marked reduction can be seen in the error rate. At 15 meters distance this trend continues and we still see a reduction of around 50% error rate. In an industrial environment, it can be see that there is a marked improvement in the number of successfully received packets when using a HP module as the Coordinator. Overall for both the LP and HP modules the PER is between 0.078% and 0.475% which is still acceptable and the communication can be considered reliable at all three separation distances.

Part one and two - Analysis of the effects of varying the packet re-transmission (retries) on the PER using LP and HP Modules in Polymer Industrial Environment:

For each separation distance the PER test was carried out for three different retry values (0, 3 and 7). The aim was to see how this would impact on the PER using all LP modules and a HP Coordinator module. The test was repeated three times to get an average value for the PER. Table 4.5 shows the results for when using all LP modules and when using a HP Coordinator. From these results it can be seen that when LP modules are used and when the number of retries is set to the maximum seven allowed in this software the PER is almost zero at all three separation distances. When the number of retries is reduced to

three the PER is still below 0.5%. When the number of retries is set to zero, there is a significant increase in the PER from almost zero to a maximum of 7.310% at 15 meters separation distance. From this result it can be said that when using all LP modules at longer distances the PER is almost negligible when the number of retries is set to seven. It can be concluded that if the number of retries is kept at three and above we get a reliable communication link.

PER using different retries for LP and HP modules in Polymer Industrial Environment						
Distance (meters)	**PER (%) All LP**			**PER (%) HP Coordinator**		
	0 Rs	**3Rs**	**7Rs**	**0Rs**	**3Rs**	**7Rs**
5m	1.002	0.078	0.094	1.683	0.036	0.012
10m	4.401	0.299	0.013	1.665	0.175	0.310
15m	7.310	0.475	0.053	2.811	0.227	0.036

Table 4.5 PER for different retries when using all LP and a HP Coordinator modules in polymer industrial environment.

When a HP Coordinator module is used and the number of retries is set to the maximum seven, the PER is almost zero at all three separation distances. When the number of retries is reduced to three the PER is still well below 0.5%. Finally when the retries are set to zero the PER jumps to 2.81% which is a significant increase. However, comparing to the LP modules, when the retries is set to zero there is a significant reduction in the PER when using a HP Coordinator from 7.310% to 2.811%. Therefore, in the polymer industrial environment, it can be concluded that although the maximum retries of seven gives a better result but there is a possibility of this not being the case when the data packet size is doubled. Therefore using a HP module as the Coordinator with the standard three retries as recommended by the IEEE 802.15.4 standard will give optimal results in terms of PER.

4.3.3 Comparison of Home/Office and Polymer Industrial Environments

In this section a comparison of the results for the Jennic modules in the home/office and polymer industrial environments is done. First, a comparison of the RSSI parameter in both environments is carried out. Then a comparison of the PER parameter is done.

4.3.3.1 *Comparison of RSSI in Home/Office and Polymer Industrial Environment*

This part of the results shows a comparison done for the RSSI parameter, which has been measured in both the home/office and polymer industrial environments. First a comparison to evaluate the performance of all LP modules in both environments is done. The graph on the left hand side in Figure 4.13 shows the results of this comparison. It can be seen that when using all LP modules at 5 meters separation distance the RSSI in home/office environment is stronger. This can be attributed to a number of factors in the industrial environment including the μIM machine that was running at that time. The other factor was the Wi-Fi wireless network in the vicinity, which was busy, and being used by a number of users at that time. There was also the dynamic nature of the environment such as people moving around and the movement of machine parts, which could have had an effect on the Line of Sight (LoS) between the transmitter and receiver. On the other hand, in the home/office environment, the likely interferences would have been from the Wi-Fi network, satellite or microwave signals which are only switched on at certain times with very low usage. At 10m separation distance and above, the industrial environment exhibits slightly better RSS. This can be linked to the fact that in the home office environment there was a brick wall in the LoS. At a distance of 15m the industrial environment has a significantly better RSS this again can be attributed to the fact that there were two brick walls between the LoS in the home office environment whereas the industrial environment only had one wall with a large glass window in it.

A comparison to evaluate the performance of a HP Coordinator module in both environments was done and the graph on the right hand side in Figure 4.13 shows the results of this comparison. When a HP module was used as the transmitter, at 5 meters separation distance the industrial environment experiences more than 50% higher signal loss compared to the home/office environment. Again, this can be attributed to the factors mentioned earlier. The home/office environment at this distance has a strong signal as there is not much interference. At 10 meters separation distance although the home/office environment has a brick wall between the LoS, the industrial environment still

experiences a slightly higher signal loss. At 15 meters, separation distance the industrial environment still experiences a slightly higher signal loss around 4%. To conclude it can be seen that the interferences in the

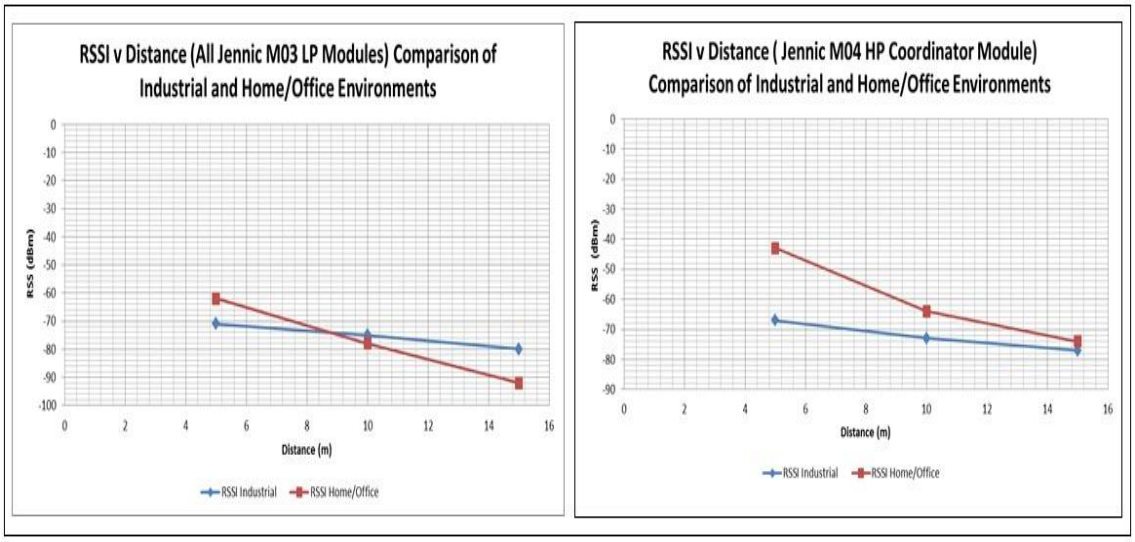


Figure 4.13 RSSI comparison for Jennic LP modules in home/office and polymer industrial environments.

industrial environment such as running machines, dynamic machine parts, people moving around and other wireless networks in the vicinity have an impact on the RSSI. The use of a HP Coordinator did not have any significant effect on the RSSI in the industrial environment although the overall RSS was slightly better than the LP modules. On the other hand, the home/office environment exhibited a significantly better RSS at shorter distances.

4.3.3.2 *Comparison of PER in Home/Office and Polymer Industrial Environment*

This part of the results shows a comparison of the PER parameter which has been measured in both the home/office and polymer industrial environments. First, a comparison to evaluate the performance of all LP modules in both environments is done and the results are shown in the left hand side graph in Figure 4.14. At 5 meters separation distance when using all LP modules the polymer industrial environment exhibits a significantly higher PER compared to the home/office environment. This trend continues at 10 and 15 meter distances. As the distance increases the PER increases in both environments, but the industrial environment exhibits almost a three times higher PER at all three separation distances.

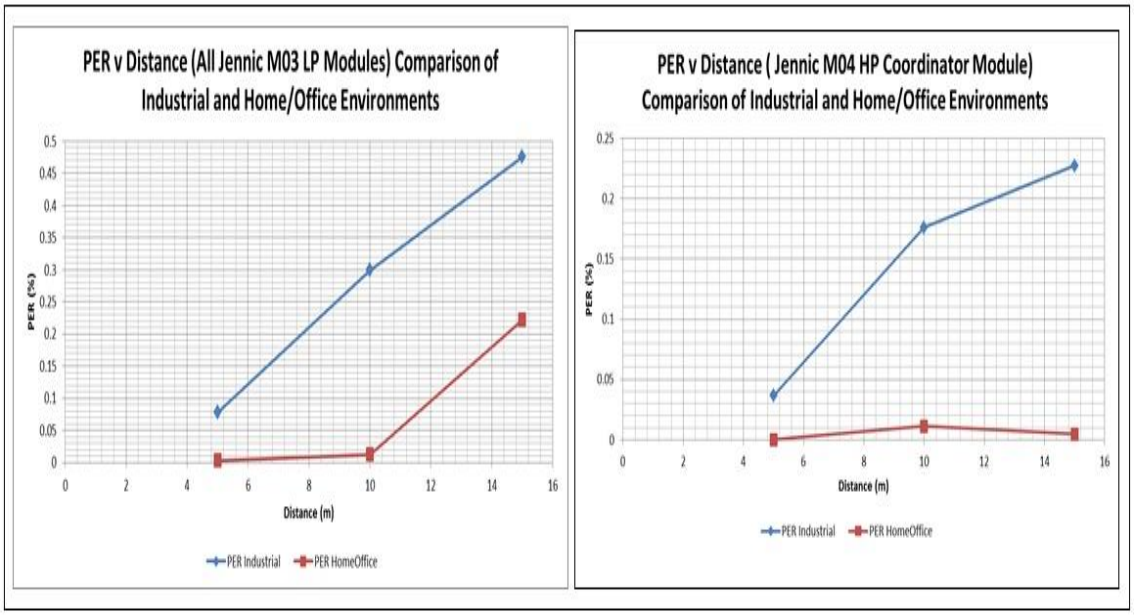


Figure 4.14 Comparison of PER using all Jennic LP modules in home/office and polymer industrial environments.

A comparison to evaluate the performance of a HP Coordinator module in both environments was done and the right hand graph in Figure 4.14 shows the results of this comparison. When using a HP module as the transmitter, the PER for the home/office environment is almost zero at all three distances hence indicating a very reliable communication link. On the other hand, the polymer industrial environment exhibits a significantly higher PER at all three distances. When a HP module is used as the transmitter, the industrial environment exhibits almost 50% less errors compared to the LP modules. Overall it can be concluded that the PER is significantly higher in the industrial environment compared to the home/office environment. This can be attributed to a number of factors including the high temperatures, moving parts, people moving around, brick walls, and the Wi-Fi network in the same vicinity. Although industrial environment experiences higher error rates due to the above-mentioned factors, the overall maximum PER using LP modules is 0.475%, which still can be considered as a very reliable communication link.

4.4 Comparison of XBee and Jennic Modules

In the previous sections both the XBee and Jennic WSN have been tested individually using the PER and RSSI parameters in both home/office and polymer industrial environments. Tests have been carried out first by using all LP modules and then swapping the LP transmitter module with a HP module. In

this section, a comparison of the performance based on PER and RSSI is done between the XBee and Jennic modules in both home/office and polymer industrial environments.

4.4.1 Comparison of RSSI for XBee and Jennic Modules

The RSSI of the XBee and Jennic modules has been compared in both the home/office and polymer industrial environments, first using all LP modules and then using a HP Coordinator as the transmitter. In Figure 4.15 the graph on the left shows the result when using all LP modules in a home/office environment. From this result it can be seen that the XBee modules exhibit on average for all separation distances about 10% better RSS. The graph at the right shows the results when using all LP modules in the polymer industrial environment. These results also show that the XBee modules experience a better RSS for all three separation distance.

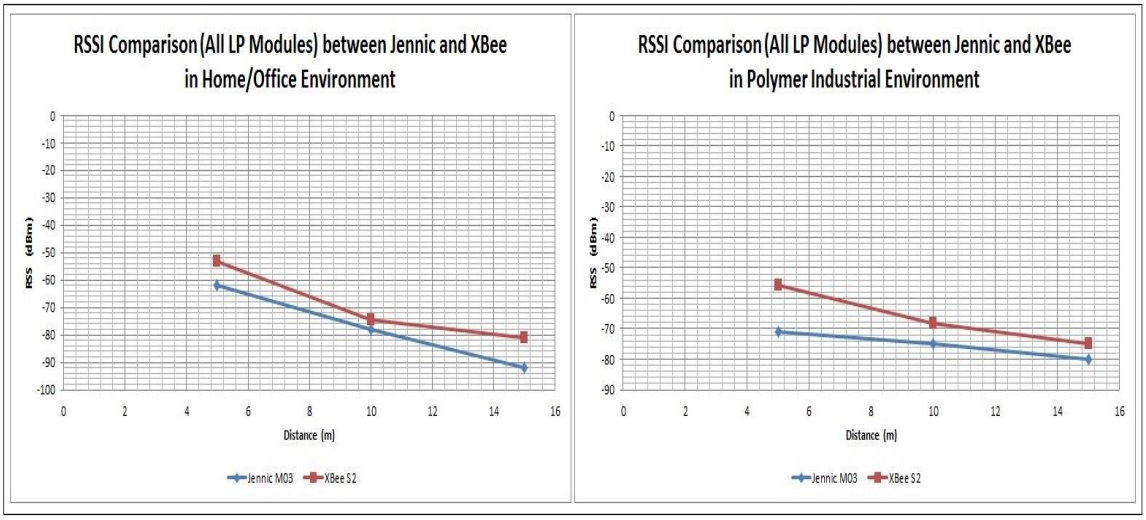


Figure 4.15 Jennic and XBee (all LP modules) RSSI comparison.

Figure 4.16 shows the comparison of the RSSI in the home/office and polymer industrial environments when using a HP Coordinator as the transmitter. The graph on the left shows the comparison result in the home/office environment. From this result, it can be seen that Jennic modules exhibit a slightly better RSS at a 5m separation distance. As the distance is increased to 10 meters and above, both the Jennic and XBee modules exhibit an almost identical RSS. The graph at the right shows the comparison results in the polymer industrial environment. From this result, it can be seen that the XBee modules experience a significantly stronger RSS at all distances when

using a HP Coordinator as the transmitter. It can be concluded from the RSSI results that the XBee modules experience a stronger RSS compared to the Jennic modules in both the environments. When using the HP module as the transmitter the XBee modules have a significanlty better RSS.

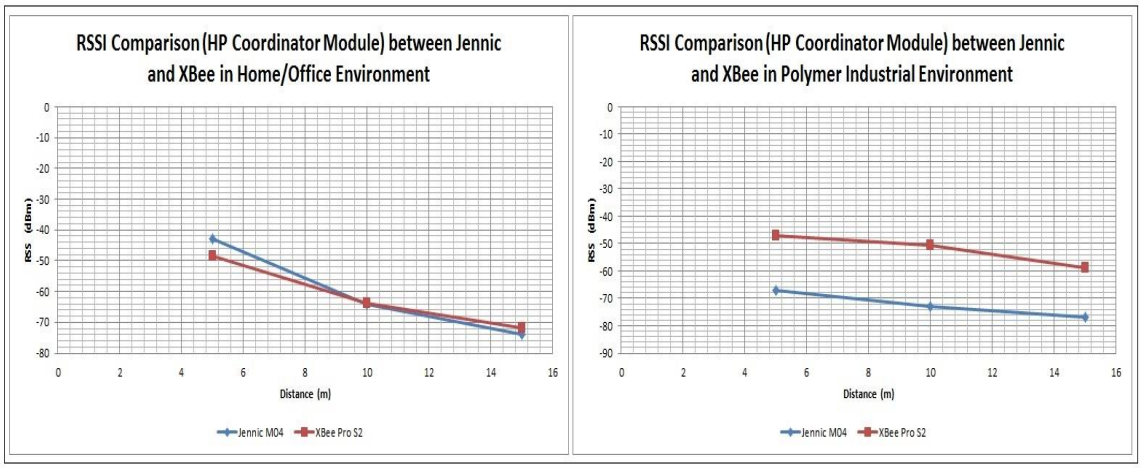


Figure 4.16 Jennic and XBee (HP module as Coordinator) RSSI comparison.

4.4.2 Comparison of PER for XBee and Jennic Modules

The PER of the XBee and Jennic modules has been compared in both the home/office and polymer industrial environments, first using all LP modules and then using a HP Coordinator as the transmitter. In Figure 4.17 the graph on the

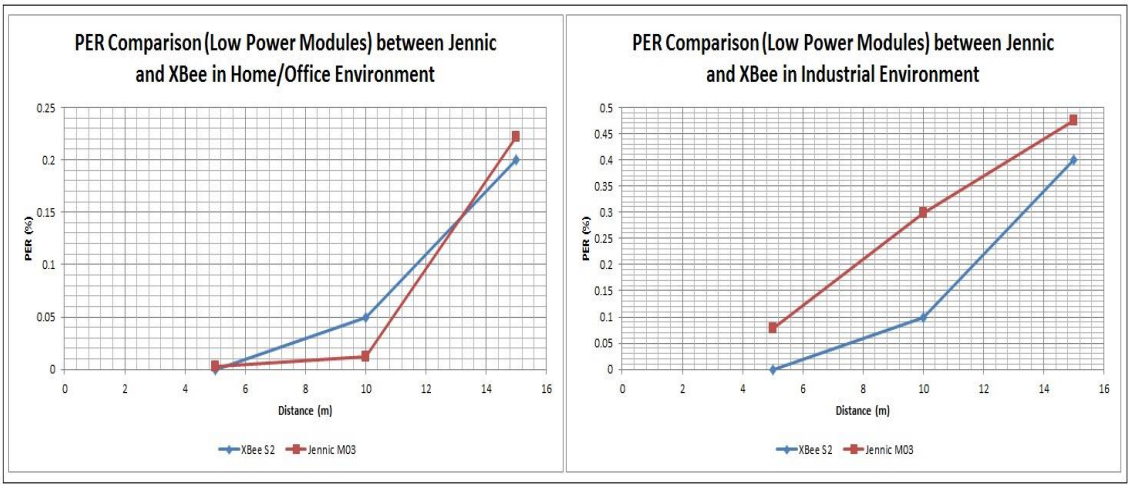


Figure 4.17 Jennic and XBee (all LP modules) PER comparison.

left shows the result when using all LP modules in a home/office environment. From this result it can be seen that the XBee LP modules have a slightly lower PER at 5 meters separation distance. At 10 meters, distance the Jennic modules PER remains almost the same whereas the XBee modules exhibit a significantly higher PER. At 15 meters separation distance the XBee modules

have a lower PER than Jennic. The graph on the right compares the PER in the polymer industrial environment when using all LP modules. It can be seen from this result that the Jennic modules have a higher PER compared to the XBee modules at all separation distances. The PER for both the XBee and Jennic modules is more than double at 10 meters and above when compared to the home/office environment. For both the Jennic and XBee modules the PER is below 0.5% at all three separation distances. Figure 4.18 shows the comparison of the PER in the home/office and polymer industrial environments when using a HP Coordinator as the transmitter. The graph on the left shows the comparison result in the home/office environment. From this result, it can be seen that at 5 meters separation distance both the XBee and Jennic modules

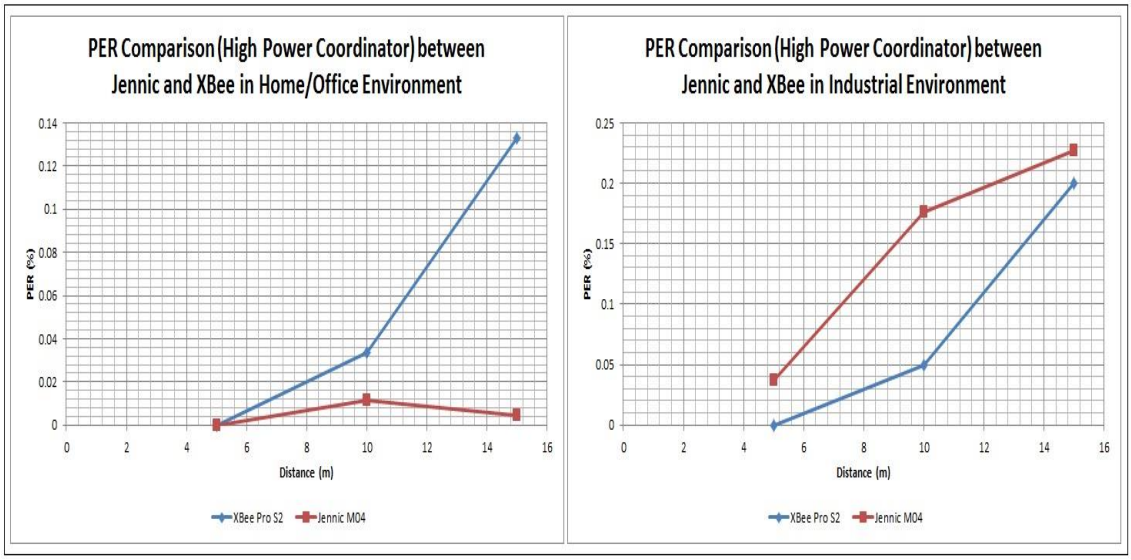


Figure 4.18 Jennic and XBee (HP module as Coordinator) PER comparison..

have zero PER. At 10 meters and above separation distance the XBee modules experience a significantly higher PER. The graph on the right shows the comparison results in the polymer industrial environment. Comparing this result with the home/office environment it can be seen that the PER in the industrial environment for the XBee modules is almost double at 10 meters and above separation distance. For the Jennic modules, the error rate is more than double at all distances. It can also be concluded from this result that the XBee modules exhibit less PER compared to the Jennic modules at all separation distances. Overall for both the XBee and Jennic modules the PER is below 0.25% at all three separation distances.

4.5 Conclusion

Although WSNs can be deployed in inaccessible or hazardous environments (inside a μIM machine with high torque electric servo drives, vibration sensors in machines and even in bearings of motors, oil pumps etc.) which are impractical with normal wired systems. Issues such as the effect of noises, co-channel interferences, multipath and other interferers on coverage area and reliability of data in these industrial environments pose a major challenge. The signal strength can be severely affected in industrial environments due to reflections off walls, floors, interferences from ISM bands, noise generated from equipment's or heavy machines and obstacles made of conductors like metals, steels, cast iron, copper etc. (Werb et al., 2005). The data integrity needs to be maintained even in the presence of noises and interferences for mission-critical data such as alarms. To ensure reliable delivery of messages from sensors to the controller node or host applications, changes in the signal level and radio connectivity need to be addressed through the careful analysis of RSSI and PER parameters (Low et al., 2005). The PER, RSSI, and LQI parameters have been used in a number of studies to assess the communication reliability of WSN in the presence of interferences from different sources. As to date there was no study using these parameters to evaluate and compare the reliability and performance of Jennic and XBee WSNs in the home/office and industrial environments.

In this chapter, a comprehensive analysis of the RSSI and PER using Jennic and XBee modules in two distinct environments was carried out. The results have shown that the XBee modules performed slightly better in terms of the RSS and PER in the polymer industrial environment. It was also noted that in regardless of the modules used in both the home/office and polymer industrial environments the presence of walls in the environments had an impact on the RSS. The RSS improved significantly when a HP Coordinator was used.

A comparison of the XBee and Jennic modules showed that overall the XBee modules had a slightly stronger RSS in both the environments when using a HP module as the Coordinator. In terms of the PER, it could be seen that the XBee modules (all LP modules and a HP Coordinator module) had a

slightly lower PER in the industrial environment at all three separation distances. In the home/office environment at 10 meters and above the Jennic modules had a slightly better PER. However, taking into consideration that in these test the number of data packets analysed for the Jennic modules for each test was 50000 whereas for the XBee modules it was 500 packets. This was purely due to the restrictions in the software used. Therefore based on the worst case scenario results using the standard three retries the highest PER for the XBee modules was 0.4% and that for the Jennic modules was 0.475%. Based on these results there is not much difference in terms of PER between the two modules and communication reliability can be considered reliable with these results. The fact that for the Jennic modules 50000 packets were analysed for each test and the PER is below 0.5% in the polymer industrial environment makes them a strong candidate for monitoring the polymer industrial environment. Also from the conclusion of chapter three, the Jennic devices were found to be very reliable as well as easy to use. Based on this conclusion the Jennic kit was chosen to be used for the monitoring of the polymer industrial environment. In the next chapter novel software will be developed to use the Jennic devices to monitor the polymer industrial environment and the μIM machines.

Chapter 5: A WSN and SOA based Process Monitoring System: LVWSM

5.1 Introduction

In the conventional and µIM fields, the ability to monitor and analyse the whole process has been one of the main focus of research for the past few years. The continuous monitoring of key process parameters has been pivotal for the overall process optimisation and control. From section 3.2 in conventional injection moulding the most common process parameters monitored include temperature and pressure; in particular in the cavity (Chen et al., 2009). The µIM process on the other hand does come with additional challenges as there are some extra process parameters related to the control of piston movement and melt storage barrel settings (Zhao et al., 2001, Whiteside, 2006, Greener and Wimberger-Friedl, 2006).

The monitoring of the µIM process has historically been limited to the µIM machine level were the data from the process is only available through the machine's embedded controller interface. Customised process monitoring systems have been developed in a number of studies (Greener and Wimberger-Friedl, 2006, Zhao et al., 2007). In these systems, the common setup has been the use of data acquisition hardware connected to a computer, which processes, displays and saves the data. The use of proprietary software and hardware that requires specialist knowledge and input from the manufacturers makes these systems difficult to integrate or interoperate with other systems and business components of the enterprise.

Whilst all the research focus has been on the monitoring of the IM/µIM processes, researchers in the industrial environment field have been looking at ways to integrate the machinery and devices in the dynamic harsh industrial environment of the factory shop floor with other business operations of the enterprise. Most of the research has focused on integrating wireless sensor nodes on the factory shop floor with the internet and enterprise level application (Guinard and Trifa, 2009b, Spiess et al., 2009a, Leguay et al., 2008). Other efforts most notably the SOCRADES project (de Souza et al., 2008) focuses on

how to integrate various devices on the factory shop floor with the enterprise infrastructure. The most common and extensively researched approach to integrate WSNs and other devices through the use of web standards has been the Service Orientated Architecture (SOA), an architectural concept which separates functions into distinct units called services (Erl, 2009). So far there has been no focus on integrating the IM/μIM processes with other business components of the enterprise or making parts of the process available to the outside world. This would bring huge benefits by allowing the real-time data to be made available on the internet or to be integrated with various external manufacturing processes and systems including: Statistical Process Control (SPC), Failure Modes and Effects Analysis (FMEA), Intelligent Process Optimisation (IPO), Process analytical Technology (PAT) etc. Such systems using this real-time data will have numerous benefits in terms of process monitoring, material/product quality and traceability alongside the usual benefits offered by Enterprise systems such as resource management, order tracking and billing.

In this chapter the architecture, design, development, and implementation of the LabVIEW Wireless Sensor Monitor (LVWSM) μIM process monitoring system is presented. This system is implemented on a central gateway and takes advantage of the Service Orientated Architecture (SOA) (De-facto solution in Enterprise-IT systems) and applies it to a network of wireless sensor nodes installed in the μIM machine and the industrial micro-moulding environment. The WSN monitors the industrial environment and the machine environment. Process data from environment and the various stages of the μIM process is transmitted wirelessly using sensor nodes to a central gateway, which converts and exposes the data as a service using Web Services. This chapter focuses on two distinct parts of the system: WSN hardware programming and interface; and the LVWSM μIM process monitoring system.

This chapter is organised as follows: Section 5.2 presents the WSN firmware development and implementation. The interface hardware is presented which is used to interface a sensor node to the Battenfeld Microsystem 50 μIM machine is presented in section 5.3. Section 5.4 presents the novel architecture

design of the LVWSM process monitoring system. Section 5.5 presents the software implementation of the LVWSM process monitoring system. In section 5.6 the developed system is tested in terms of its functionality, GUIs, data collection and web services and the results from these tests are presented. Section 5.7 discusses issues related to the system performance and implementation. Finally, section 5.8 presents the conclusion of this chapter.

5.2 WSN Programming and Machine Interface

To monitor the polymer industrial environment and the machines in this environment, sensors need to be installed in the environment and the machine to collect and relay the data back to a central point where it can be analysed and processed. To achieve this, a network of wireless sensor nodes was installed in the Battenfeld Microsystem 50 µIM machine and the industrial micro-moulding environment of the Polymer MNT Laboratory at the University of Bradford.

5.2.1 Wireless Sensor Nodes and Configuration

The Jennic JN5148 WSN kit (Jennic Ltd, 2010b) tested in chapters three and four and shown in Figure 5.1 was used for this purpose. From the results in chapter four for reliability the WSN had the following setup:

The Coordinator node in the WSN uses the DR1047 controller board and a Jennic M04 High Power (HP) module set at standard three retries and was connected to a central gateway computer. The End Device sensor nodes in the WSN used a DR1048 sensor board with a Jennic M03 Low Power (LP) module. The Jennic M03 LP module had the following specifications.

- 32-bit Enhanced CPU with 128K of RAM and ROM.
- A 4 channel 12 bit Analogue to Digital Convertor (ADC), Two 12 bit Digital to Analogue Convertors (DAC) and two comparators
- 21 Digital I/O ports
- 2.4GHz IEEE802.15.4 compliant integrated radio.
- Powered using two “AAA” batteries or using a 5V mains adaptor.

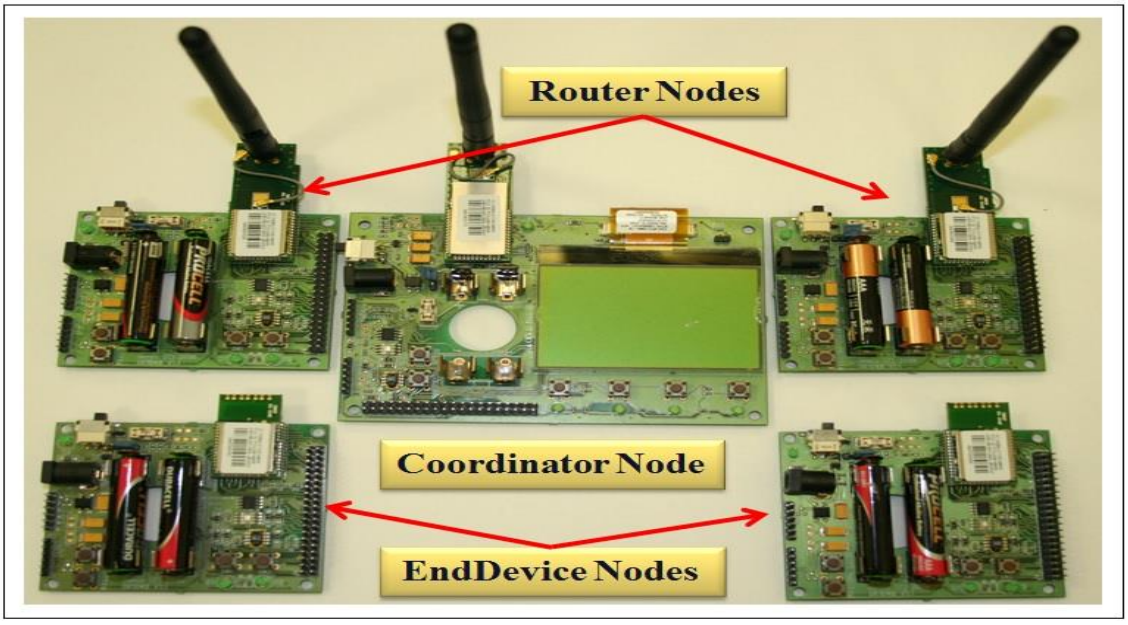


Figure 5.1 The Jennic WSN node types.

Whereas the Jennic M04 HP module had the same specifications as the LP modules except for the HP radio with more power and longer range. The sensor boards used for the End Device nodes contain an on-board Sensirion SHT11 single-chip multi-sensor module for the measurement of temperature and relative humidity. The device includes two calibrated micro-sensors for temperature and relative humidity, which are coupled to a 14-bit analogue-to-digital converter and a serial interface circuit. Conversion may be programmed with 8-, 12- or 14-bit accuracy, depending on application resolution or speed requirements. Humidity is measurable over 0-100% relative humidity and temperature over the range –40°C to 85°C (Jennic Ltd, 2010a). The board also had a TAOS TSL2550 light sensor, which was not used in this work. The controller board used for the Coordinator node also had the same on-board sensors as well as additional features including an on-board LCD unit.

5.2.2 WSN Node Requirements and Programming

The monitoring of the polymer industrial environment and the machines using a WSN requires nodes with different firmware configurations. In order to program these nodes accordingly a list of requirements needs to be identified.

5.2.2.1 *Environmental Node Requirements*

In section 3.2, some of the most common key parameters that have been used in previous research to monitor the plastics industry environment were

identified. The polymer industrial environment required the monitoring of ambient temperature and humidity. To fulfil these requirements a sensor node is required which has the ability to monitor these parameters. The Jennics DR1048 sensor board described in section 5.2.1 has an on-board Sensirion SHT11 single-chip multi-sensor module for the measurement of temperature and relative humidity, which can be used for this purpose. Therefore, the firmware for this node will be programmed to take these requirements into consideration and modified accordingly.

5.2.2.2 *Machine Node Requirements*

Also from chapter three section 3.2 at the machine level the key parameters identified for monitoring in a µIM process are:

f) Cavity Pressure (high-resolution data)
g) Nozzle Pressure (high-resolution data)
h) Temperature (high-resolution data)
i) Piston Displacement
j) Piston Velocity

To fulfil these requirements a sensor node is required which has the ability to monitor these parameters. The on-board sensors from Jennic can only monitor the temperature and humidity. Therefore, external sensor inputs are required for the pressure, displacement and velocity. The Battenfeld MicroSystem 50 µIM machine described in Appendix B has high precision external and integrated sensors installed to monitor the above key parameters. A Tempsonic RH-Series magnetostricitve displacement transducer measures the piston displacement and velocity. The cavity pressure is measured using a Dynisco 4005 piezoelectric force transducer and the nozzle pressure is measured using the Battenfeld µIM machines integrated pressure transducer (need to verify name and spec). The temperature is measured using a K-type thermocouple (need exact name and spec). The output from these sensors is amplified using Kistler charge amplifiers giving a 10V output signal. In order to monitor these outputs using a sensor node from Jennic, the sensor board's analogue inputs and on-board ADC will need to be utilised to acquire the analogue outputs form the machine and convert them to a digital form. Therefore, the firmware for the

machine node will need to be modified to take the above requirements and changes into consideration.

5.2.2.3 *WSN programming*

The Eclipse IDE based Jennics tool-chain (Jennic Ltd, 2010b) was used to modify the firmware code for each node.

Coordinator Node: The firmware for the Coordinator node connected to a central gateway via serial/USB connection has the following features:

- Configures its own on-board Humidity and Temperature sensors
- At the network level
  - the Co-ordinator Selects the frequency channel to be used by the network (usually the one with the least detected activity)
  - Starts the network
  - Allows nodes to join the network
  - Provide message routing and other services
- Sends its own and connected nodes data to the Universal Asynchronous Receiver/Transmitter (UART) port

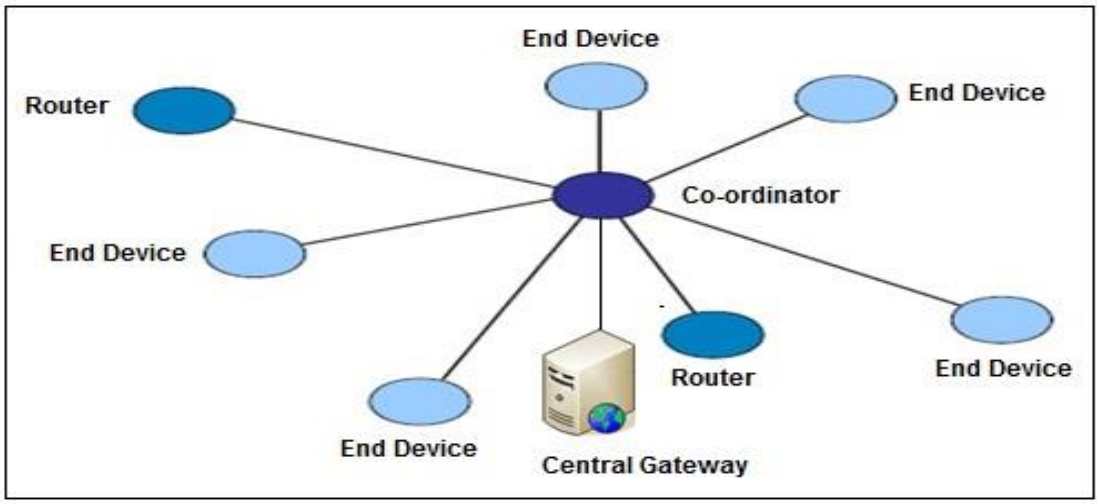


Figure 5.2 The star WSN topology.

The JenNET WSN supports end-to-end, Star or Mesh topologies. In this work the star topology shown in Figure 5.2 was adopted in which a Coordinator node attached to the a central gateway acts as the central point to which Routers and End Device nodes can connect to.

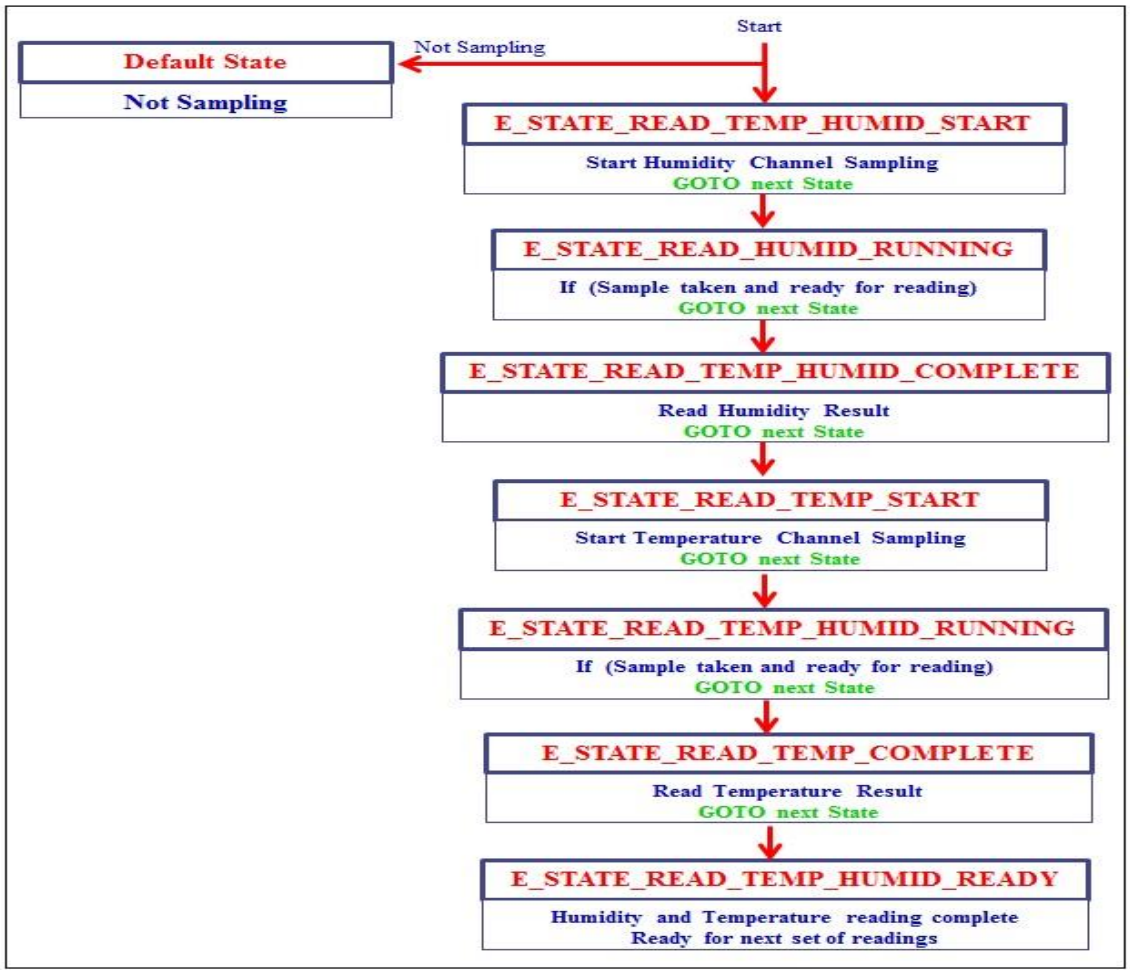


Figure 5.3 The state machine for reading the on-board humidity and temperature sensors.

Router Node: One Router node was used to ensure that the two End Device nodes data is passed onto the Coordinator node. The firmware for the Routernode allows it to initially join the network created by the Co-ordinator. The Router initialises and takes readings sequentially from its on-board temperature and humidity sensors as well as the supply voltage. It also allows further sensor boards to join the network. The Router takes readings from its sensors every one second, and outputs the readings and network address to its UART (19200-8-N-1). The Router's readings are also sent to the Co-ordinator through the network. After every set of readings, the Router pauses for one second before taking the next reading. During this pause period, the Router continues to run the JenNet stack to allow other sensor boards to send their data to the Co-ordinator.

Environmental End Device Node: One of the End Device nodes was used to monitor the temperature and humidity of the polymer industrial environment. The firmware for this node allows it to initially join the network formed by the Co-ordinator and Routers. The End Device initialises and takes readings sequentially from its on-board temperature and humidity sensors as well as the supply voltage. A state machine was implemented to read the sensor readings as shown in Figure 5.3. The implementation code for this state machine is shown in Figure 5.4. The state machine starts by sampling the humidity channel. It then goes into the next state in which it waits for the sampling to be completed and ready for reading. Once this is completed, it goes into the next state in which it reads the humidity result. It then moves onto the temperature sensor and goes through a similar sequence as before until it reads the temperature reading. The node also uses the ADC to read the supply voltage (battery or mains). Measurement is performed using a state machine similar to the one described above to ensure that it never blocks. The node sensor and supply voltage readings along with the node address are output to its UART (19200-8-N-1) as well as being sent to the Co-ordinator through the network. Once a set of readings has been taken, the End Device enters sleep mode for one second before waking to take the next set of readings. Being in sleep mode allows the End Device to conserve power when not taking readings or using the radio.

The second End Device node was used to monitor the Battenfeld Microsystem 50 µIM machine process parameters. The firmware for this node allows it to initially join the network formed by the Co-ordinator and Routers. The End Device initialises and takes readings from its on-board temperature and humidity sensors. To monitor the µIM machine process parameters a state machine shown in Figure 5.5 was implemented to read the analogue ports and convert the analogue signal into digital form using the on-board ADC. The implementation code for this state machine is shown in Figure 5.6. The nodes ADC channels are read sequentially using the state machine. When the sampling begins, the state machine starts with channel ADC1 and goes into the E_STATE_READ_ADC_START state. In this state the ADC1 channel is set to busy and the common parameters for all on-chip analogue resources are set

through the vADC_Config() function. The vJPI_AnalogueStartSample() function is used to start the ADC sampling. Theoretically a 12bit ADC should convert an analogue signal into 2^{12} discrete values that is 4096 discrete values.

```
PRIVATE void vSensor_ReadTempHumidity(void)
{
   switch(eStateTempHumid)
{
case E_STATE_READ_TEMP_HUMID_START:
   vHTSstartReadHumidity();
eStateTempHumid = E_STATE_READ_HUMID_RUNNING;
    break;

case E_STATE_READ_HUMID_RUNNING:
if((u32JPI_DioReadInput() & HTS_DATA_DIO_BIT_MASK) == 0)
{
eStateTempHumid = E_STATE_READ_TEMP_HUMID_COMPLETE;
}
break;

case E_STATE_READ_TEMP_HUMID_COMPLETE:
sSensorReading.u16Humid  = u16HTSreadHumidityResult();
eStateTempHumid = E_STATE_READ_TEMP_START;
break;

case E_STATE_READ_TEMP_START:
   vHTSstartReadTemp();
eStateTempHumid = E_STATE_READ_TEMP_HUMID_RUNNING;
break;

case E_STATE_READ_TEMP_HUMID_RUNNING:
if((u32JPI_DioReadInput() & HTS_DATA_DIO_BIT_MASK) == 0)
{
eStateTempHumid = E_STATE_READ_TEMP_COMPLETE;
}
break;

case E_STATE_READ_TEMP_COMPLETE:
sSensorReading.u16Temp = u16HTSreadTempResult();
eStateTempHumid = E_STATE_READ_TEMP_HUMID_READY;
break;

case E_STATE_READ_TEMP_HUMID_READY:
break;

default:
break;
}
}
```

Figure 5.4 The implemented code for reading the on-board humidity and temperature sensors.

Machine End Device Node: band-gap voltage of 2.4V as a reference voltage. Therefore, a 1-bit change represents a voltage change of 586μV (2.4V/4096). After the conversion, a value of 0V should theoretically give a 0 digital output and the full 2.4V should give a value of approximately 4096. The state machine moves to the E_STATE_READ_ADC_CONVERTING state in which it tries to ascertain if the conversion has completed through a polling function. Once the sampling has been completed, it then moves into the E_STATE_READ_ADC_COMPLETE state in which (for the relevant channel) it

calculates the bit change using the resolution and internal reference voltage. Once the change has been calculated the ADC channel is reset to not busy.

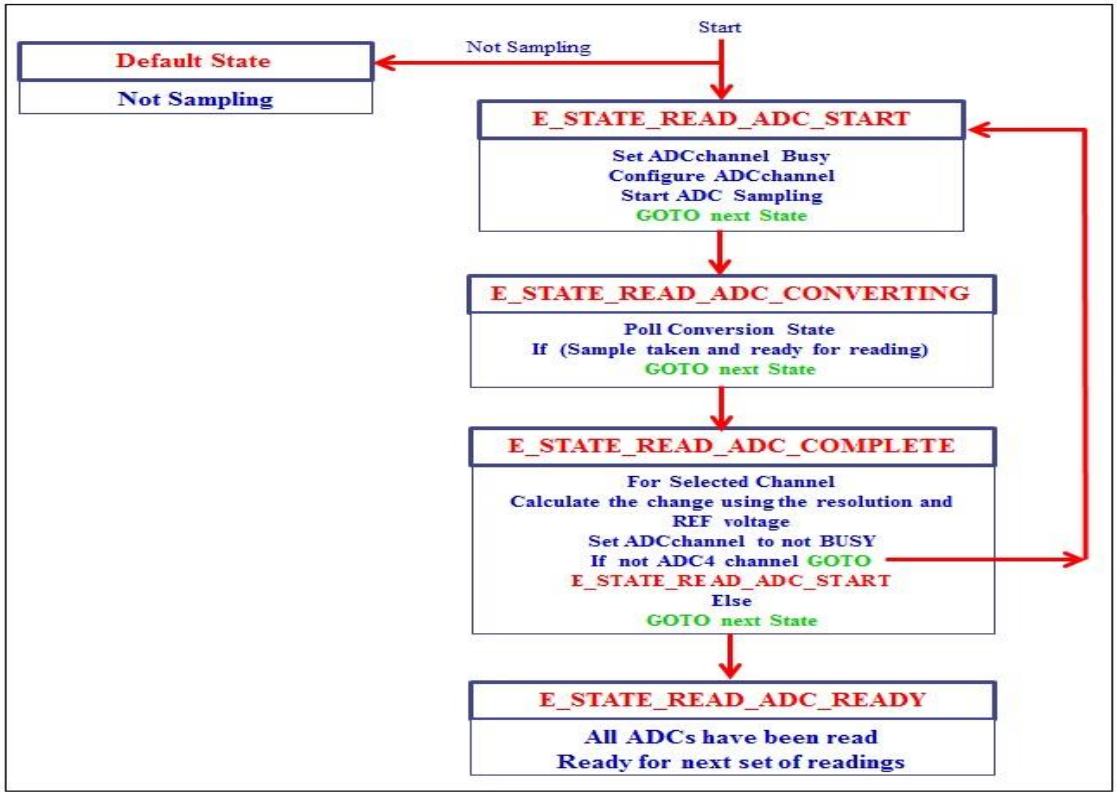


Figure 5.5 The state machine for sequentially reading the four on-board ADC Channels.

The state machine then goes back to the beginning to sample the next channel. It continues until it has completed sampling of all four analogue channels. The node also uses the ADC to read the supply voltage (battery or mains). Measurement is performed using a state machine similar to the one described for the on-board sensors in Figure 5.4 to ensure that it never blocks. The node outputs the on-board sensor, supply voltage, and the set of ADC channel readings along with its own address to its UART (19200-8-N-1) as well as sending to the Co-ordinator through the network. Once a set of readings has been taken, the End Device enters sleep mode for one second before waking to take the next set of readings. Being in sleep mode allows the End Device to conserve power when not taking readings or using the radio. The implemented code on each node is compiled and the resultant binary files are uploaded onto the nodes using the general methodology for programming a node firmware (Appendix A.1).

```
/*****************************************************************
* NAME: vADC
* Added By Umar Raza
* DESCRIPTION:
* Read Each ADC channel input sequentially. Reading is performed using a
* state machine to ensure that it never blocks. 16/02/2011
*
*****************************************************************
PRIVATE void vESensor_ReadADC(uint8 ADCchannel) {
uint16 u16ADC1Reading;
uint16 u16ADC2Reading;
uint16 u16ADC3Reading;
uint16 u16ADC4Reading;
/*vPrintf("\n In vESensor_ReadADC");*/
switch (eStateADC) {
case E_STATE_READ_ADC_START:

switch (ADCchannel) {
case E_JPI_ADC_SRC_ADC_1:
ADC1Busy=1;
break;
case E_JPI_ADC_SRC_ADC_2:
ADC2Busy=1;
break;
case E_JPI_ADC_SRC_ADC_3:
ADC3Busy=1;
break;
case E_JPI_ADC_SRC_ADC_4:
ADC4Busy=1;
break;
}

vADC_Config(ADCchannel); /*Single shot sampling */
vJPI_AnalogueStartSample();
eStateADC=E_STATE_READ_ADC_CONVERTING;
/*vPrintf("\n ADC Start OK");*/
break;

case E_STATE_READ_ADC_CONVERTING:
/*vPrintf("\n Before ADC Polling");*/
if(!bJPI_AdcPoll()) {
eStateADC=E_STATE_READ_ADC_COMPLETE;
/*  vPrintf("\n ADC Converting OK");*/
}
break;
case E_STATE_READ_ADC_COMPLETE:
if(ADCchannel==E_JPI_ADC_SRC_ADC_1) {
u16ADC1Reading=u16JPI_AnalogueAdcRead();

/* Input range is 0 to 2.4V. ADC has full scale range of 12 bits.
 Therefore a 1 bit change represents a voltage of approx 586uV */
sSensorReading.u16ADC1 =((uint32)((uint32)(u16ADC1Reading * 58
+((uint32)(u16ADC1Reading * 586)>>1)))/1000;

vJPI_AnalogueDisable(ADCchannel);
ADC1Busy=0;
eStateADC=E_STATE_READ_ADC_START;
/*vPrintf("\n ADC1 Complete OK");*/
}
if(ADCchannel==E_JPI_ADC_SRC_ADC_2) {
u16ADC2Reading=u16JPI_AnalogueAdcRead();

/* Input range is 0 to 2.4V. ADC has full scale range of 12 bits.
 Therefore a 1 bit change represents a voltage of approx 586uV */
sSensorReading.u16ADC2 =((uint32)((uint32)(u16ADC2Reading * 586)
+((uint32)(u16ADC2Reading * 586)>>1)))/1000;

vJPI_AnalogueDisable(ADCchannel);
ADC2Busy=0;
eStateADC=E_STATE_READ_ADC_START;
/*vPrintf("\n ADC2 Complete OK");*/
}
if(ADCchannel==E_JPI_ADC_SRC_ADC_3) {
u16ADC3Reading=u16JPI_AnalogueAdcRead();

/* Input range is 0 to 2.4V. ADC has full scale range of 12 bits.
 Therefore a 1 bit change represents a voltage of approx 586uV */
sSensorReading.u16ADC3 =((uint32)((uint32)(u16ADC3Reading * 586)
+((uint32)(u16ADC3Reading * 586)>>1)))/1000;
vJPI_AnalogueDisable(ADCchannel);
ADC3Busy=0;
eStateADC=E_STATE_READ_ADC_START;
/*vPrintf("\n ADC3 Complete OK");*/
}
if(ADCchannel==E_JPI_ADC_SRC_ADC_4) {
u16ADC4Reading=u16JPI_AnalogueAdcRead();

/* Input range is 0 to 2.4V. ADC has full scale range of 12 bits.
 Therefore a 1 bit change represents a voltage of approx 586uV*/

sSensorReading.u16ADC4 =((uint32)((uint32)(u16ADC4Reading * 586)
+((uint32)(u16ADC4Reading * 586)>>1)))/1000;
vJPI_AnalogueDisable(ADCchannel);
ADC4Busy=0;
eStateADC=E_STATE_READ_ADC_READY;
/*vPrintf("\n ADC4 Complete OK");*/
}
break;
case E_STATE_READ_ADC_READY:
/*vPrintf("\n ALL ADCs Have Been Read");*/
break;
default:
/*vPrintf("\n NOT Sampling ADC channels yet");*/
break;
}
}
```

Figure 5.6 The implemented code for sequentially reading the four on-board ADC Channels.

5.3 Hardware Interface Design: WSN Layout and Testing

The output from the machine sensors was amplified using Kistler charge amplifiers giving a 10V output signal. The maximum voltage allowed on the sensor boards pins is 2.5V and interfacing this directly to the outputs from the machine sensors would damage the sensor nodes on-board circuitry. Therefore, an interface circuitry was required to convert the 10V outputs to

2.5V. A simple hardware interface using a voltage divider circuit was made for this purpose. Using a 10K and 30K resistor in the voltage divider circuit gave an output of approx. 2.5V. Figure 5.7 shows the designed machine interface circuit connected to the machine sensors using BNC connectors and on the other side, it is connected to wireless sensor node using a 40-pin ribbon cable connector.

The wireless machine sensor node reads the four analogue data inputs from the µIM machine along with the ambient temperature and humidity. The analogue inputs are converted to a digital form by using the wireless sensor

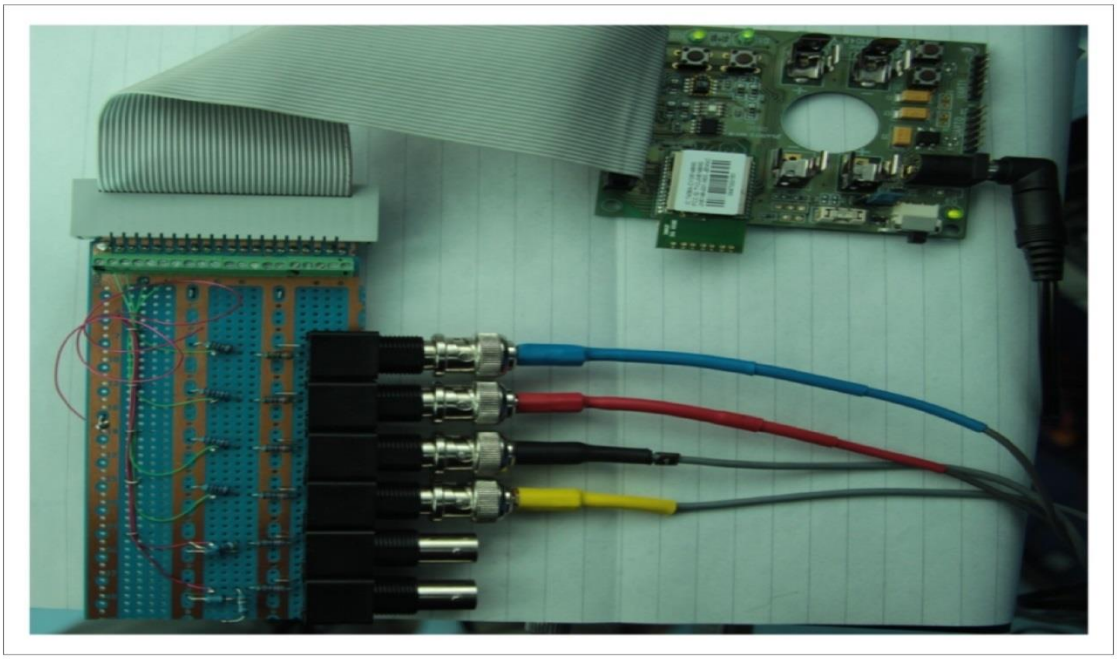


Figure 5.7 The machine interface circuit connected to a sensor node and machine outputs.

nodes on-board 12bit ADC. The digital data is sent wirelessly to the coordinator node in the micro-moulding lab connected to a central gateway laptop via RS232/USB link at a baud rate of 115200 Kb/s. The coordinator node processes this data by identifying the address of the node, which sent it and then makes it available on the UART port.

5.3.1 µIM Process Monitoring WSN Layout

The WSN was setup in the Polymer MNT Laboratory as shown in Figure 5.8. Two LP environmental End Device nodes were installed in the Polymer MNT Laboratory at a distance of approximately 10 meters from the Coordinator to monitor the temperature and humidity of the industrial environment while the LP

machine End Device node was connected to a Battenfeld Microsystem 50 µIM machine via the interface described in the previous section to monitor the process data and machine environment. The distance between the machine node and the Coordinator was approximately 5 meters. One LP Router node was also installed at approximately 5 meters distance from the Coordinator to ensure that the two End Device nodes data is passed onto the Coordinator node. The two Environmental nodes were installed in the clean rooms to monitor the conditions in those environments and were used in later experiments.

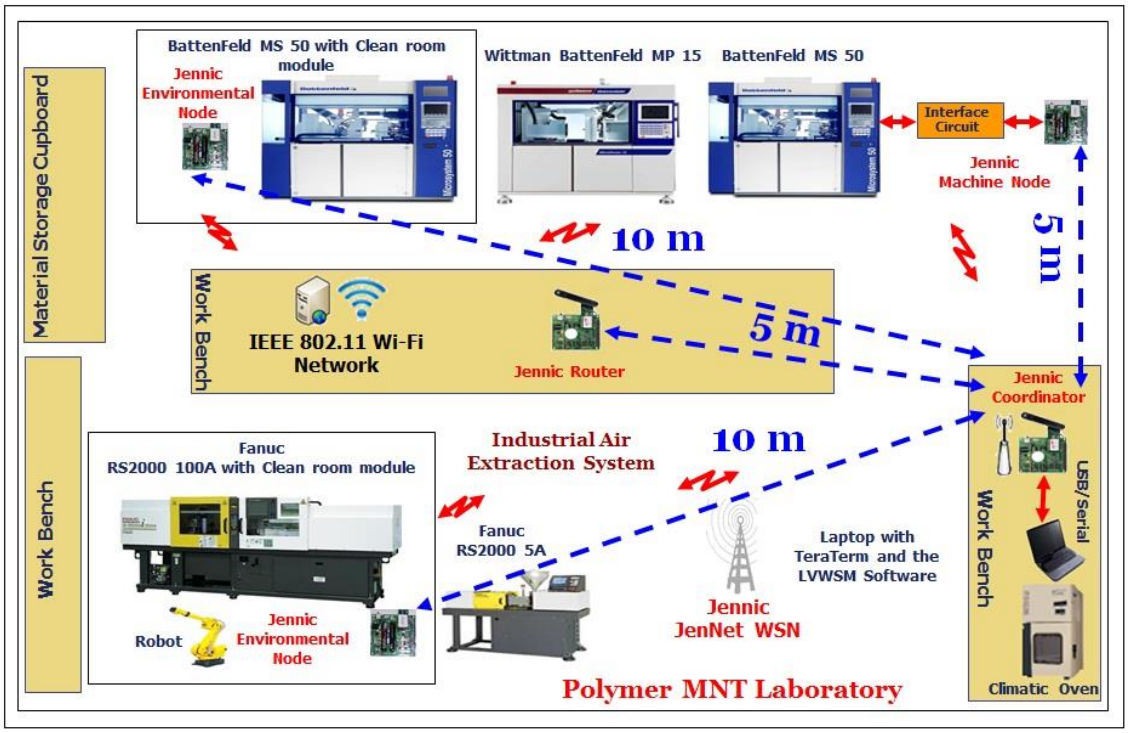


Figure 5.8 Process monitoring WSN layout.

5.3.2 µIM Process Monitoring WSN Testing

With the firmware loaded onto the nodes and the WSN configured and setup as shown in Figure 5.8, the network had to be tested to verify that the modifications made to the firmware function correctly. This was achieved by using a terminal utility program to read the serial port of the computer to which the Coordinator node is connected. An off-the-shelf open source terminal utility called Teraterm (Teranishi, 2010) was used for this purpose. The WSN was started and the data from the network was collected using the terminal utility program. Figure 5.9 shows the sensor data values displayed in the Teraterm utility. From this, it

could be seen that the sensor nodes were transmitting the data to the controller and the output is as expected. The environmental sensor node with the address “0x158d00:0x7e39b” was just displaying the on-board sensor readings whereas the machine node with the address “0x158d00:0x110955” was displaying the on-board sensor and the ADC channel readings. The controller node with the address “Local” was also displaying its own on-board sensor readings. The Teraterm utility was fine for just capturing the raw data direct from the serial port but to monitor a µIM process using this utility would not be sufficient.

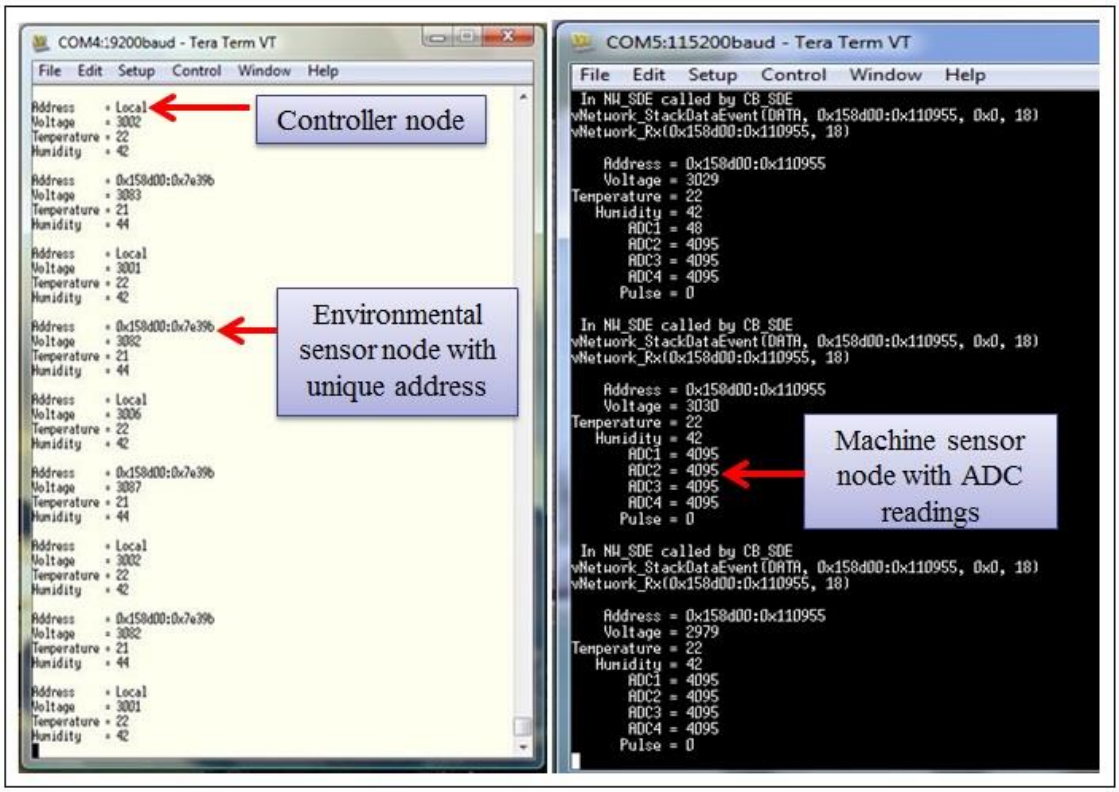


Figure 5.9 Sensor data displayed using the Teraterm utility.

Therefore in order to overcome the limitations with a terminal utility program the other option was to develop a novel application which would read data in real-time direct from the serial port of the PC to which the Coordinator node is connected to. The application would then process, format, display and convert this data into web services allowing it to be integrated with web-based applications. To achieve this, the LVWSM µIM process monitoring system was developed using National Instruments LABVIEW development environment (National Instruments, 2010).

5.4 LabVIEW Wireless Sensor Monitor (LVWSM): Design

The monitoring of the µIM process has historically been limited to the use of the µIM machine embedded controller data that is only available through the machines LCD interface. The use of proprietary software and hardware that requires specialist knowledge and input from the manufacturers makes these systems difficult to integrate or interoperate with other systems and business components of the enterprise. So far, there has been no focus on integrating the IM/µIM processes with other business components of the enterprise or making parts of the process available to the outside world. This would bring huge benefits by allowing the real-time data to be made available on the internet or to external software. In this section, a new process monitoring system called the LVWSM that addresses the above shortcomings is presented. This system implements the SOA concept using Web Services and applies it to the WSN described earlier, hence allowing the remote and internet monitoring of some the key µIM process parameters.

First, the system requirements are presented followed by the complete architecture and design. Next, the system software that was developed using the LabVIEW 2010 environment is described. This is followed by testing the developed system by using it in the calibration process of the nodes on-board sensors. Next, the clean room facility is monitored using the LVWSM system. The system is then further tested for its data logging ability by collecting the data from the environment and the Battenfeld Microsystem 50 µIM machine. Finally, the systems Web Service implementation part is tested for its ability to be consumed by web-based applications. The implementation is shown through two scenarios: First using LabVIEW Web UI and then using Presto Web Mashup builder.

5.4.1 Process Monitoring Software Requirements

In order to determine what the functional requirements of the LabVIEW based process monitoring system would be, a simple analysis of the Jennic WSN and the format of the sensor network data are required. Figure 5.10 gives an overview of the Jennic WSN and the format of the data, which it writes to the serial port. The data is collected by the End Device sensor nodes and transmitted to the Coordinator node either directly or via the Router node.

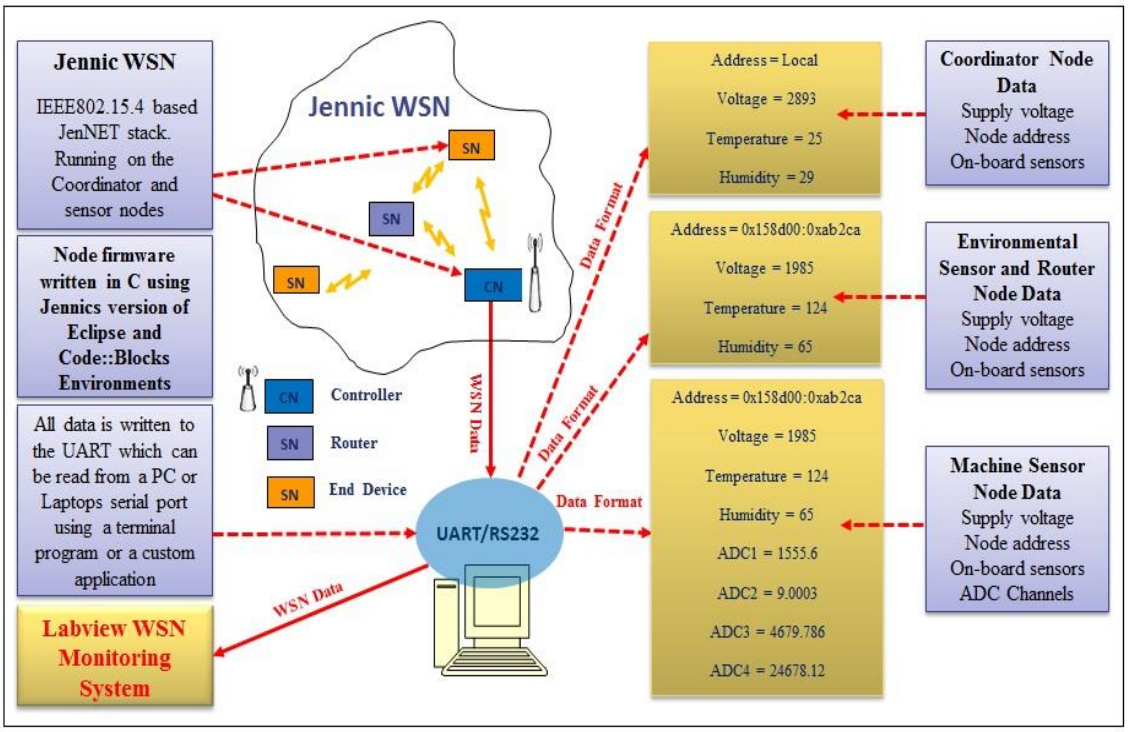


Figure 5.10 An overview of the Jennic WSN and the node data format.

The Coordinator node then writes all the data to the UART every second. This data can then be read by either serial terminal software or a custom application. Analysing Figure 5.10 the system to be developed in LabVIEW would need to have the following functionality.

- Read the USB/Serial port of the Laptop/PC
- Store the read data in a buffer in order to avoid data loss
- Parse the data and separate each node and the relevant sensor data
- Write the data for each node in a separate file with date and time stamp
- Write the data for each node to a relational database with data and time stamp
- For each node plot the sensor data in a graph format in real-time.
- Automatically detect when a new node has joined the network
- Convert the data from the machine and environment into a web services allowing it to be integrated and consumed by web-based applications and other components of an enterprise.

5.4.2 LVWSM System Architecture Design

To develop a system with the above functional requirements would involve the creation of software components for each individual requirement. Once this is achieved, these components would then need to be integrated to make them work together as a unified system. To allow the WSN data to be consumed by different web and enterprise based applications there is a need to share and exchange this data in a common message format. To realise this vision, an LVWSM architecture based on the web services concept is proposed. Using web service technology enables SOA, which when applied can solve the integration aspects of the architecture. In LabVIEW, the SOA is realised using Restful web services over standard HTTP protocol. If a LabVIEW program (VI) output needs to be converted into a Web Service, an interface needs to be developed using LabVIEW. Once this is done then LabVIEW creates a web service and publishes the interface and access information onto to the local NI service name registry (a little network service running on the local computer).

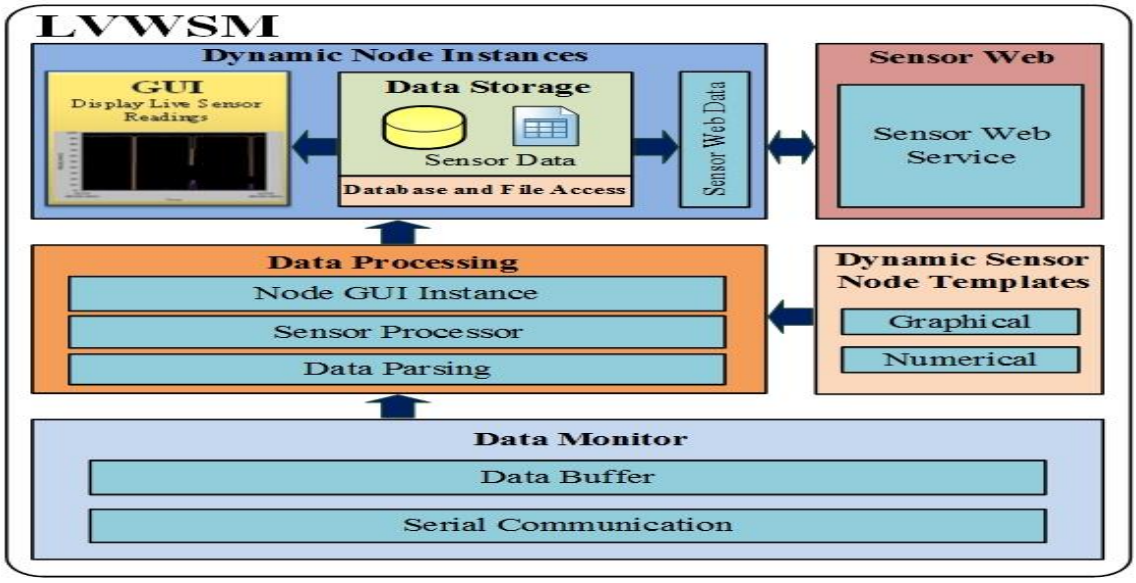


Figure 5.11 LVWSM System Architecture Design.

The web service clients can locates entries in the service registry using various find operations and then binds to the service provider in order to invoke one of the web services. Multiple services can be accessed if the service provides multiple services. Figure 5.11 illustrates the proposed LVWSM architecture made up of five layers; Data Monitor layer, Data Processing layer, Dynamic Sensor Node Templates layer, Dynamic Node Instances layer, and Sensor Web layer.

5.4.2.1 *Data Monitor Layer Design*

The main task of the data monitor layer is to monitor and communicate with the Serial/UART port for any data. Figure 5.12 shows the functional architecture of the Data Monitor layer in more detail. This layer has two components; Serial Communication and Data Buffer. The Serial Communication component will read the Serial/UART port for the WSN data written to it by the Coordinator node. Each node in WSN sends its data every second to the Coordinator, which writes it to the UART. Therefore, in order to ensure that there is no data loss, this component needs to read the serial port every half a second and store the retrieved data. The data retrieved from the serial port by Serial Communication component is stored in the Data Buffer component.

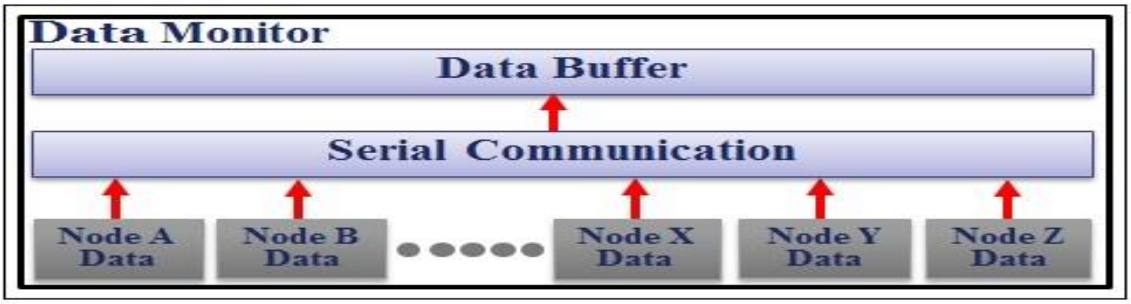


Figure 5.12 Data Monitor layer functional architecture design.

The Data Buffer needs to have either a variable size or be big enough to handle large amounts of data. As the data is written to the serial port every second, therefore the Data Buffer component will use up its storage space very quickly. The storage space required will depend upon the number of sensor nodes connected to the network. Since each sensor node transmits 32 bytes of information, therefore a buffer size of approximately 4Kb would be sufficient to support up to 128 devices. In order to ensure that the buffer does not overflow, the data needs to be retrieved from this buffer slightly quicker than the time it takes to write. Therefore, time synchronization is required between the writing functions in the Serial Communication component and the reading functions used to read data from the Data Buffer component. In LabVIEW the concept of data queues as the name suggests, queue data. This data is stored in a buffer in memory until it is read from the queue so nothing is lost (National Instruments, 2012e). The Data queue concept will be used in the Data Buffer component to ensure that data is not lost.

5.4.2.2 *Data Processing Layer Design*

The Data Processing layer is at the heart of the process monitoring system as it is responsible for the processing of live data from the buffer; formatting data for storage, further conversion and display. This layer utilises an advanced programming concept of dynamic application creation in LabVIEW; which allows the creation of instances of an application at runtime based on a defined template. This concept is adapted for this system because of the dynamic nature of the WSN. Figure 5.13 illustrates the functional architecture of the Data Processing layer in detail. This layer has two main components; Data Parser and Sensor Processor. The Sensor Processor has five sub components; Node Selector, Environmental Node, Machine Node, Node Instance Creator and Node Instance Update.

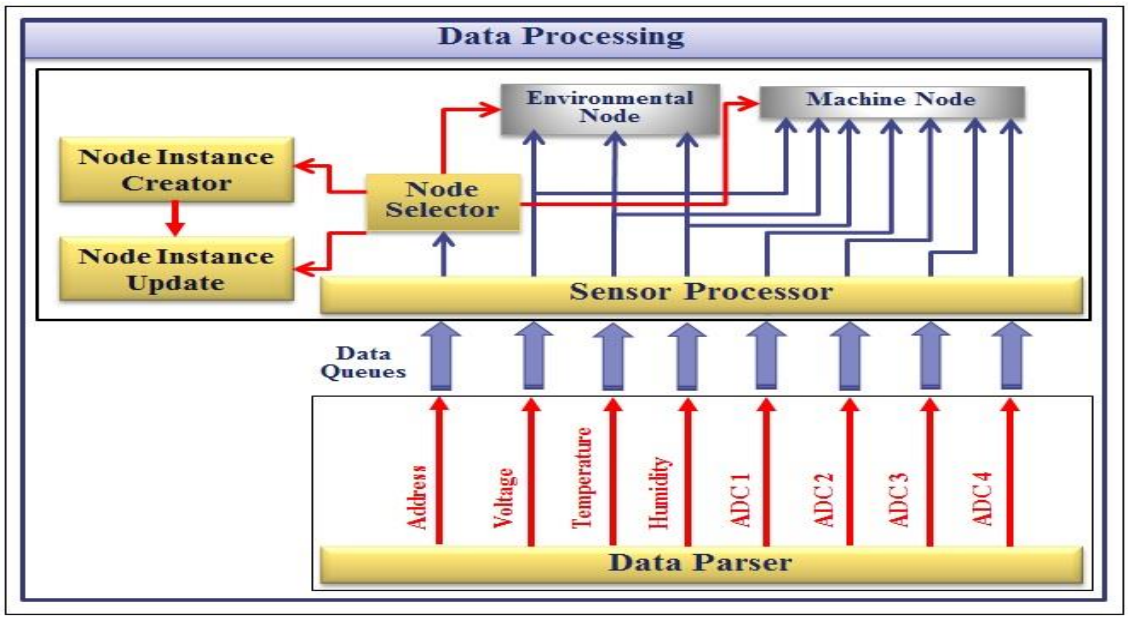


Figure 5.13 Data processing layer functional architecture design.

5.4.2.2.1 **Data Parser Design**

The Data Parser component retrieves the data from the Data Buffer component of the Data monitor layer. The format of the retrieved data is the same as the one illustrated in Figure 5.10. The Data Parser component compares the retrieved data with data tokens created from the format shown in Figure 5.10. The comparison, results in the detection of the node information (address, voltage, sensor name and readings, ADC channel name and readings) from the continuous data stream. Separate data queues are used to store the detected

node data. Data queues have been used to ensure data reliability when passing information to the next component in the system

5.4.2.2.2 Sensor Processor Design

The Sensor Processor component is responsible for retrieving the node data from the separate data queues in a synchronised manner. The data is further formatted by this component and forwarded to the Node Selector sub component.

Node Selector: The Node Selector sub component analyses the data received from the main Sensor Processor component. It first analysis the node address to ascertain whether this node is new or if it already exists in the node registry. Based on the result it will either create and select a new entry in the node registry or update and select an existing node entry. The subsequent data is marked to be associated with the selected node until the next set of data is received. It is also further analysed to classify if this data is from a machine sensor node or environmental sensor node and based on the result it forwards it either to the Environmental Node sub component or to Machine Node sub component.

Environmental Node: This sub component formats the data by adding information to indicate the type of node that will receive the data sent by the environmental sensor node and allows it to be used by a dynamic application instance creator and application template. The concept of dynamic applications in LabVIEW allows the creation of instances of an application at runtime based on a defined template. It makes the data available for the Node Instance Creator and Node Instance Update sub components.

Machine Node: This sub component is similar to the environmental node sub component. It formats the data sent by the machine node to be used by a dynamic application instance creator and application template. It makes the data available for the Node Instance Creator and Node Instance Update sub components.

Node Instance Creator: This sub component uses the data provided by the Environmental and Machine Node sub components. It initialises and starts

the dynamic application and depending upon the type of node, it waits for the Dynamic Sensor Node Templates layer to provide a template for that particular node. Once the template is loaded, an instance of an application is created in a new window. The Dynamic Node Instances layer handles the running of this instance.

Node Instance Update: This sub component uses the data provided by the Environmental and Machine Node sub components to update existing application instances. It formats and sends the data to the Dynamic Node Instances layer.

5.4.2.3 *Dynamic Sensor Node Templates Design*

This layer provides the templates to the Data Processing layers Node Instance Creator sub component. Figure 5.14 illustrates the functional architecture of this layer. It consists of a number of application templates. These templates were created based on the sensor nodes outputs. For the environmental node two templates were created graphical indicator version and numerical indicator version. Both are described and illustrated in more detail in Figure 5.31 and Figure 5.32 of the LVWSM software implementation section. For the machine node a single graphical template was created which is described and illustrated in more detail in Figure 5.33 of the LVWSM software implementation section. Depending upon which node is selected in the Data Process layer the relevant template is sent to the Node Instance Creator component. This use of dynamic applications with custom templates makes this architecture scalable and it gives the ability to cater for different types of nodes as well as WSNs from different vendors.

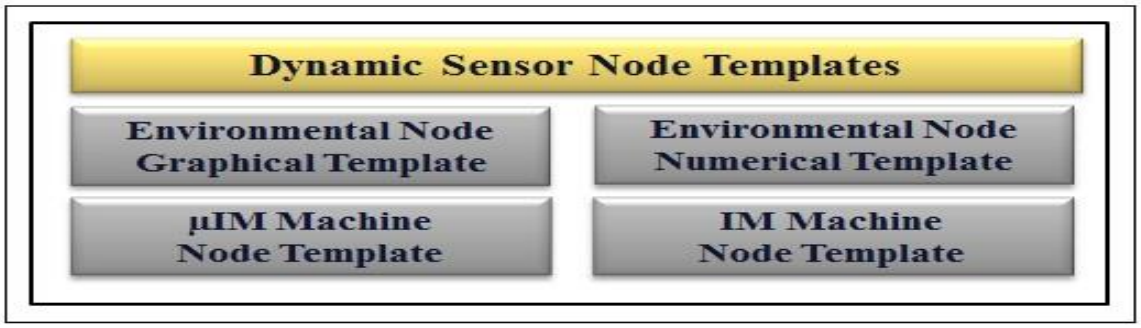


Figure 5.14 Dynamic Sensor Node Templates layer functional architecture design.

5.4.2.4 *Dynamic Node Instances Layer Design*

The Dynamic Node Instances layer is responsible for the running of the dynamic applications and consists of three components; Data storage, Display Graph, and Sensor Web Data. Figure 5.15 illustrates the functional architecture of the Dynamic Node Instances layer in detail.

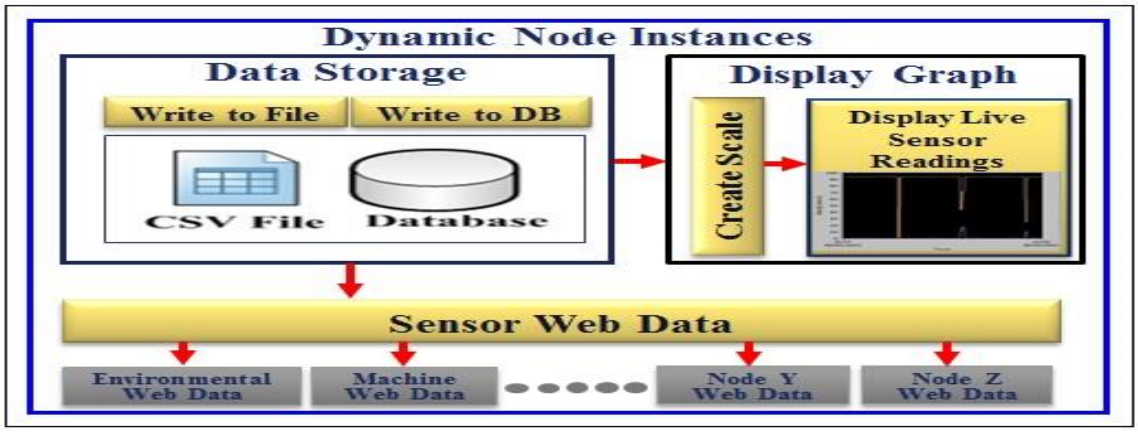


Figure 5.15 Dynamic Node Instances layer functional architecture design.

5.4.2.4.1 **Data Storage**

The Data Storage component has the primary task of formatting the data by appending extra information for logging. It uses two sub components; Write to File and Write to DB to implement the primary function.

Write to File: The Write to File sub component has three main tasks. The first task is to check if the data is from a new node using the node address and name. If this is true, it creates a new Comma Separated Values (CSV) file in a default logging directory; otherwise, it searches for the file using the node address and name. If the file is found it opens it for writing else it throws an error informs the user that the file does not exist and if a new one should be created. The second task of this sub component is to format the data to be logged in by creating comma separated columns with the relevant headings. It also adds a date stamp column to show the date and time of the sensor reading. In the third task, if the node is new then it will write the data to the file from the beginning otherwise, it appends the data to the end of the file indicating to the user that the data was logged more than once. In either case, this sub component continues to write data until a command is received to stop in which case it closes the file.

Write to DB: The Write to DB sub component has the primary function of logging the data to a relational database. The MySQL database (Oracle Corporation, 2012) was used for this purpose and the dynamic application connects to it when the user selects to log data into a database. The first task is to verify if a database name exists and if does not, then a new database is created using the node name. It then creates a new table using the sensor names as columns and an extra column for the date and time stamp. Otherwise, it searches for the database and opens it for writing. It then searches for the table and appends the values to the end of the table. This sub component continues to write data until a command is received to stop in which case it closes the database.

5.4.2.4.2 Display Graph Design

The Display Graph component has the primary task of plotting the live sensor data, which gets updated every second. It achieves this by first updating each sensor reading with calibration settings calculated in a test carried out in section 5.6.5.1 of this chapter. The calibrated reading is then sent to the LabVIEW's waveform chart component, which plot each sensor input against time. This component has the ability to allow the users to view each sensor plot separately or together. It achieves this using a sub component called Create Scale, which allows the switching between different sensor plots. The sub component has the task of changing and updating the sensor plot view and coordinate scales at runtime. This is achieved by monitoring event changes such as the clicking of a button; depending upon which button is clicked it changes the view as well as updating the XY scales accordingly.

5.4.2.4.3 Sensor Web Data Design

The main task of the Sensor Web Data component is to make each sensor nodes data available to the Sensor Web layer that is used by the Web Services created for each node. It runs in the background waiting for any requests from the Sensor Web layer to provide the latest readings for a particular node. Once a request is received, it identifies which node readings are being requested by analysing the incoming request message for the node address. The data for the identified node is then made available to the Sensor Web layer using shared variables.

5.4.2.5 *Sensor Web*

The primary task of the Sensor Web layer is to create Web Services for each sensor node that utilise the data provided by the Dynamic Node Instances layer (through its Sensor Web Data component). This layer consists of one main

Figure 5.16 Sensor Web layer functional architecture.

component the Sensor Web Service and three sub components; Create WS, Deploy WS, and URL Mappings. Figure 5.16 illustrates the functional architecture of the Sensor Web layer in detail. The Web Services created in LabVIEW are implemented using Restful Web Services (Richardson and Ruby, 2007). The RESTful web services apply Roy Fieldings REST design principles (Fielding, 2000) to develop web services. The REST design principle is the underlying architecture style of the World Wide Web hence applying REST principles means direct integration with the Web. RESTful Web services focus on the Web resources as the main abstraction. The LabVIEW Web Services (National Instruments, 2012c) created by this layer are hosted on a LabVIEW Web Server deployed on the same machine as the LabVIEW runtime. The LabVIEW Web Server was configured using instructions provided by National Instruments (National Instruments, 2012a).

5.4.2.5.1 **Create WS**

The main task of this component is to create and build Web Services for each node in the WSN and this is achieved in five steps. The first step is to create a Web Method application in LabVIEW, which contains the functionality of the Web Service and accepts requests from Web clients. Based on the Web client request the Web Method is executed using the inputs as arguments. The second step is to create the actual Web Service using the node names. In this step, the Web Service is linked to the Web Method. The third step is to configure the Uniform Resource Locator (URL) Mappings in other words

mapping the Web address to be used to access the Web service. In this step the format that the Web Method returns the data to the Web client is also configured. There are a number of formats, which the data can be converted into, and LabVIEW allows text, JSON, XML and HTML formats. In this application, the format is set to XML as it is the most common and standardized format endorsed by software industry market leaders. It is simple, easy to be read and understood. The fourth step is to select the Hyper Terminal Transport Protocol (HTTP) method to be used to access the Web Method. In the final step, the Web Service is built and is ready to be deployed on Web Server.

5.4.2.5.2 Deploy WS

Once the Web Service has been created this component is used to deploy the Web Service onto the LabVIEW Web server running on the local machine. This is achieved by using LabVIEW's deploy method described in (National Instruments, 2012a). Once the Web Service is deployed, it can be accessed using the defined URL mappings from any standard web browser.

5.4.2.5.3 Sensor Web Service

The Sensor Web Service component is responsible for organising the Web Services as well as handling requests from the Web clients through the URL Mappings component. Depending upon the client request the relevant sensor node service is invoked. The invoked service executes its functionality through its Web Method which sends a request to the Dynamic Node Instances layer (through the Sensor Web Data component) to retrieve data from the relevant node.

5.4.2.5.4 URL Mappings

The main task of this component is to take request from Web clients and compare them with the existing URL mappings. If the mapping is valid then it sends the request to the Sensor Web Service component to forward it to the relevant web service. An example of the URL mapping for the machine Web Service is http://localhost:8085/machineWS/110955. Executing this URL from a web browser returns the XML output shown in Figure 5.46 of the LVWSM Tests section. The URL mapping for the environmental Web Service is http://localhost:8085/environWS/11091a. Executing this URL from a web

browser returns the XML output shown in Figure 5.46 of the LVWSM Tests section. The Web Service URL is made up of the server name, port number, web service name and the URL mapping.

5.4.2.6 *The Complete LVWSM Functional Architecture Design*

The complete functional architecture for the LVWSM process monitoring system is shown in Figure 5.17. This diagram illustrates how the different layers of the LVWSM system described in the previous sections are linked together. It shows in more detail which components in each layer are used to communicate with components in the other layers. The Data Monitor layer continuously monitors the serial port for WSN data, which is written to the port every second. The data is stored in a buffer and is consumed by the Data Processing layer in a synchronized manner. The Data Processing layer processes the data by passing it through a data parser, which identifies individual sensor nodes by comparing the data with data tokens created from the template format shown in Figure 5.10. Once nodes have been identified, the Dynamic Node Instances layer creates an application at runtime using predefined sensor node templates provided by the Dynamic Sensor Node Templates layer. For every new node joining the WSN, a new application is created at runtime which uses templates selected based on the type of node. The sensor node application created at runtime has the following functionalities:

- Automatically detects when a new node has joined the network
- Creates a new application for the new node at runtime
- Provides various sensor node templates
- Plots the sensor data in a graph format in real-time. Logs the data for each node in a separate file with date and time stamp
- Logs the data for each node to a relational database with data and time stamp
- Converts the data from the machine and environment into a web services allowing it to be integrated and consumed by web-based applications and other components of an enterprise.

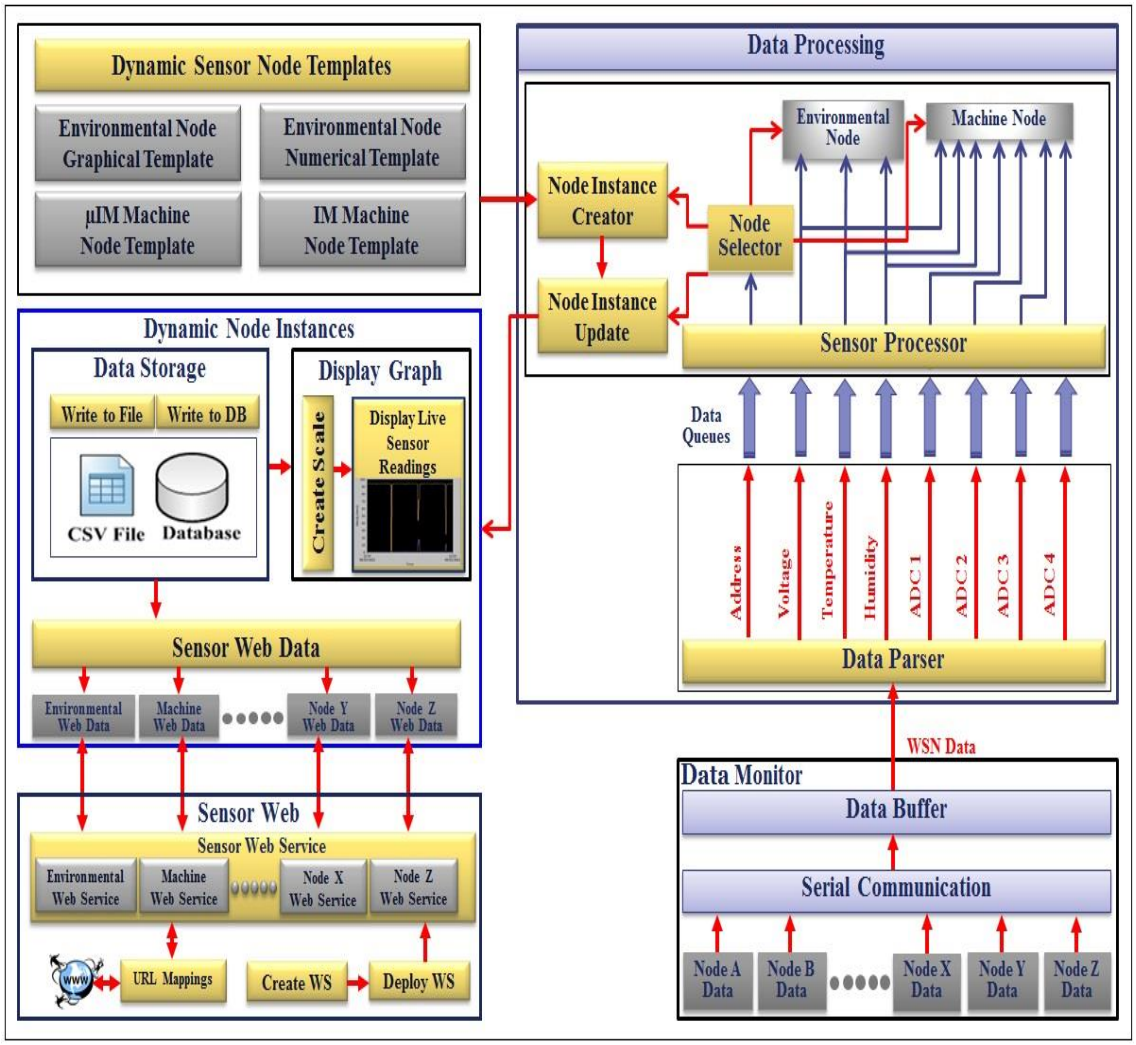


Figure 5.17 The LVWSM functional architecture design.

5.5 LVWSM Software Implementation

The LVWSM µIM process monitoring system described in the previous section was implemented using the National Instruments LABVIEW 2010 (Teranishi, 2010, National Instruments, 2010) software environment. This section describes the design and development of the software components using the LabVIEW environment. The LabVIEW environment is a visual programming language, uses graphical user interfaces (called front panels), and links them into the development cycle via the graphical source code called block diagram. Programs written in LabVIEW are called virtual instruments (VIs) and each VI has three components: the block diagram, the front panel and a connector panel. The front panel serves as the user interface and uses controls and indicators to build the user interface. The controls are inputs and indicators are outputs of the user interface. The block diagram is used to develop the program

and contains all the standard programming components such as variables, controls, functions, arrays, structures etc. It is in the block diagram that the functionality of the front panel components can be programmed. The connector panel allows the VIs to be used as blocks (functions) in the block diagrams of other VIs. In this section, an overview of the software implementation for the LVWSM system is presented. Figure 5.18 shows the developed VI components for each layer of the LVWSM system. The following sections will give an overview of the developed VI components.

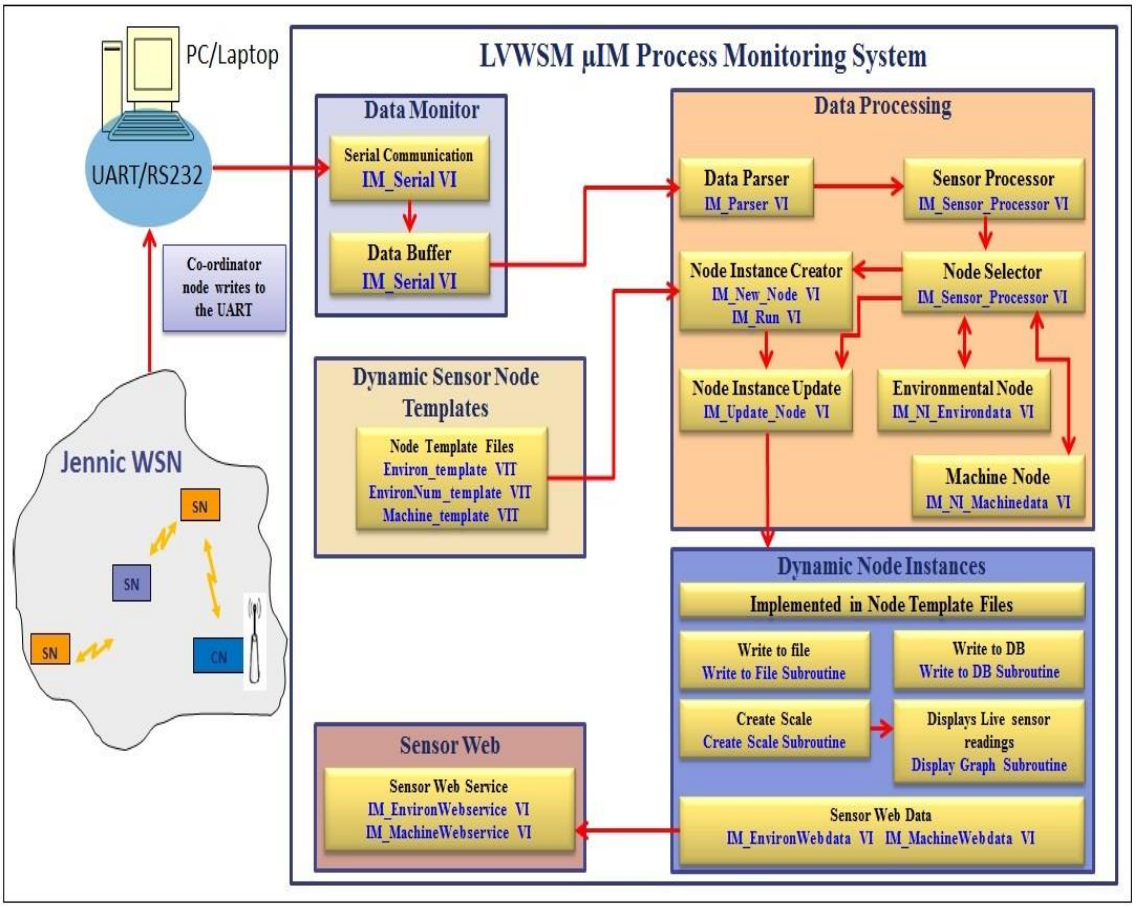


Figure 5.18 Developed VI components for each layer.

5.5.1 Data Monitor layer Implementation

The Data Monitor layer was implemented in using a single VI called the IM_Serial.VI. The Serial Communication component was implemented using LabVIEW's Virtual Instrument Software Architecture (VISA) component. The VISA is a standard I/O language for instrumentation programming and configuration. It has the ability to control Serial, Ethernet, USB and other LabVIEW based instruments by making appropriate driver calls depending on

the type of instrument being used (National Instruments, 2012b). The Serial Communication component was implemented as shown in Figure 5.19. The VI starts by first initialising the Serial port using the VISA Configure Serial Port

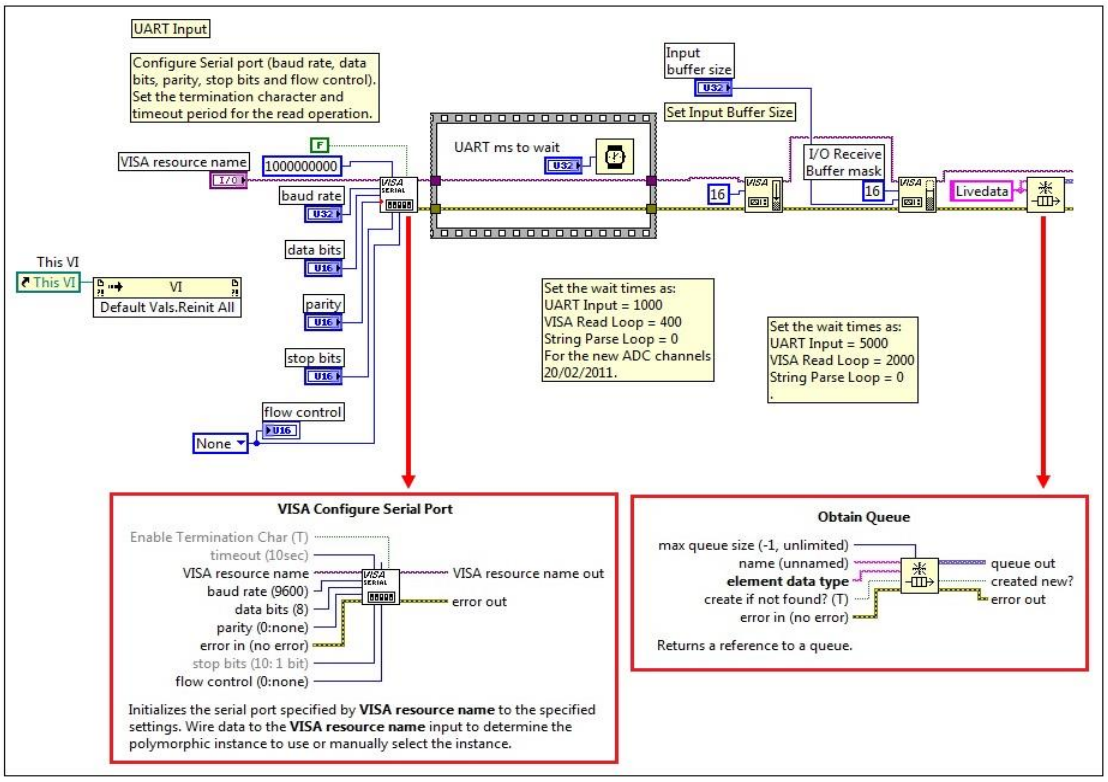


Figure 5.19 LabVIEW implementation of the Serial component.

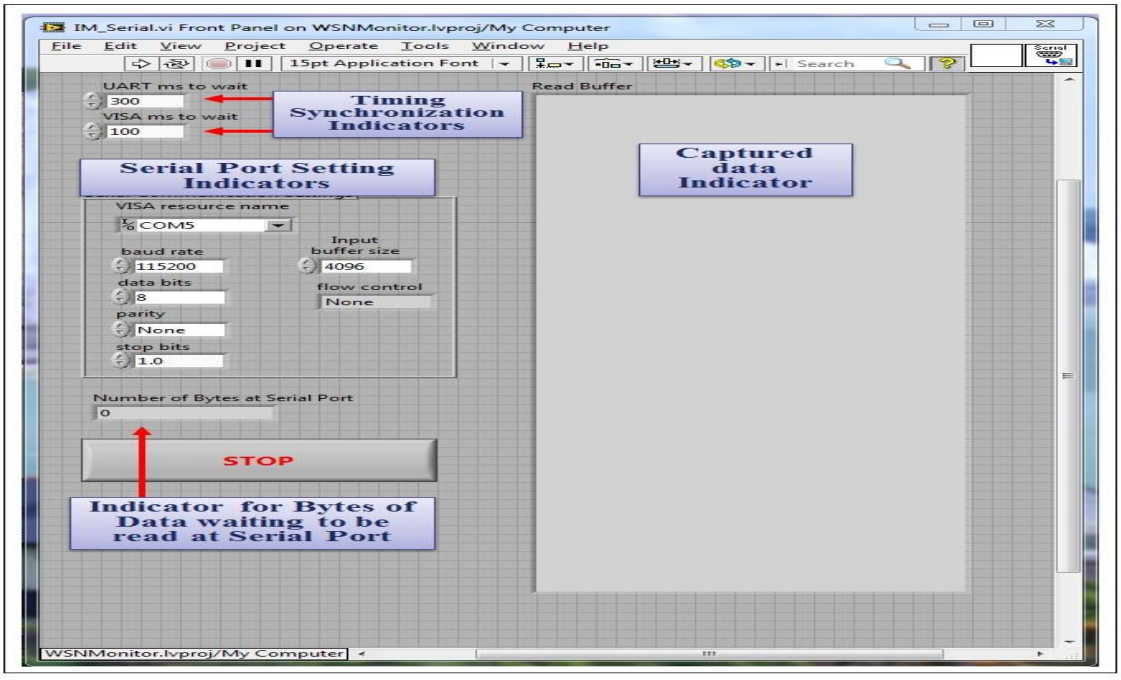


Figure 5.20 Front Panel GUI for the serial component.

function. It uses the settings provided by the user in the front panel interface shown in Figure 5.20. The VI then waits for a successful connection to be established, it then flushes the serial buffer and sets its size. Before reading the serial buffer, the Obtain Queue function is used to initialise a data queue for storing the raw data from the serial buffer. If there is no error in configuring the serial port then the serial buffer is read at a rate faster than the time it takes to write to the serial buffer using the VISA Read function. The Data Buffer component is implemented using the data queue functions. Once the data has been read, the Enqueue Element function is used to enter the data into a queue. The data in this queue is made available as an input to the IM_Sensor_Processor VI that is used to implement the Data Processing layer. The complete code for the Data Monitor layer can be found in Appendix D.1.

5.5.2 Data Processing layer Implementation

The Data Processing layer, which is at the heart of the LVWSM system was implemented in a number of files. Figure 5.18 shows the components in this layer and the files that implement each component.

5.5.2.1 *Data Parser Component Implementation*

The Data Parser component was implemented in a single VI called the IM_Parser.VI. As the format of the data written to the serial port by the

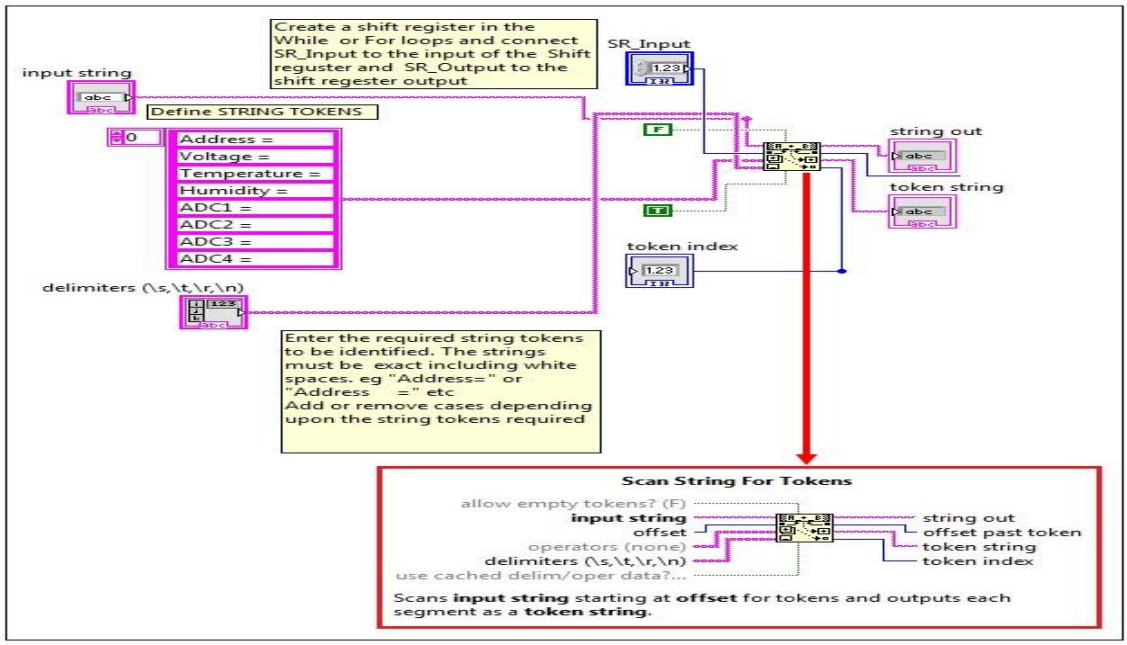


Figure 5.21 LabVIEW implementation of the Data Parser component.

Coordinator node was known beforehand, this component separated key data from the incoming stream of raw data by comparing it with defined string tokens. This was implemented by using the Scan String for Tokens function and is illustrated in Figure 5.21. Once a string is identified it is then fed into a data queue for further processing by the Sensor Processor component implemented in the IM_Sensor_Processor VI. The complete code for this component can be found in Appendix D.2.2

5.5.2.2 *Sensor Processor Component Implementation*

The Sensor Processor component is implemented in the IM_Sensor_Processor VI. This component processes the data and removes any inconsistencies like

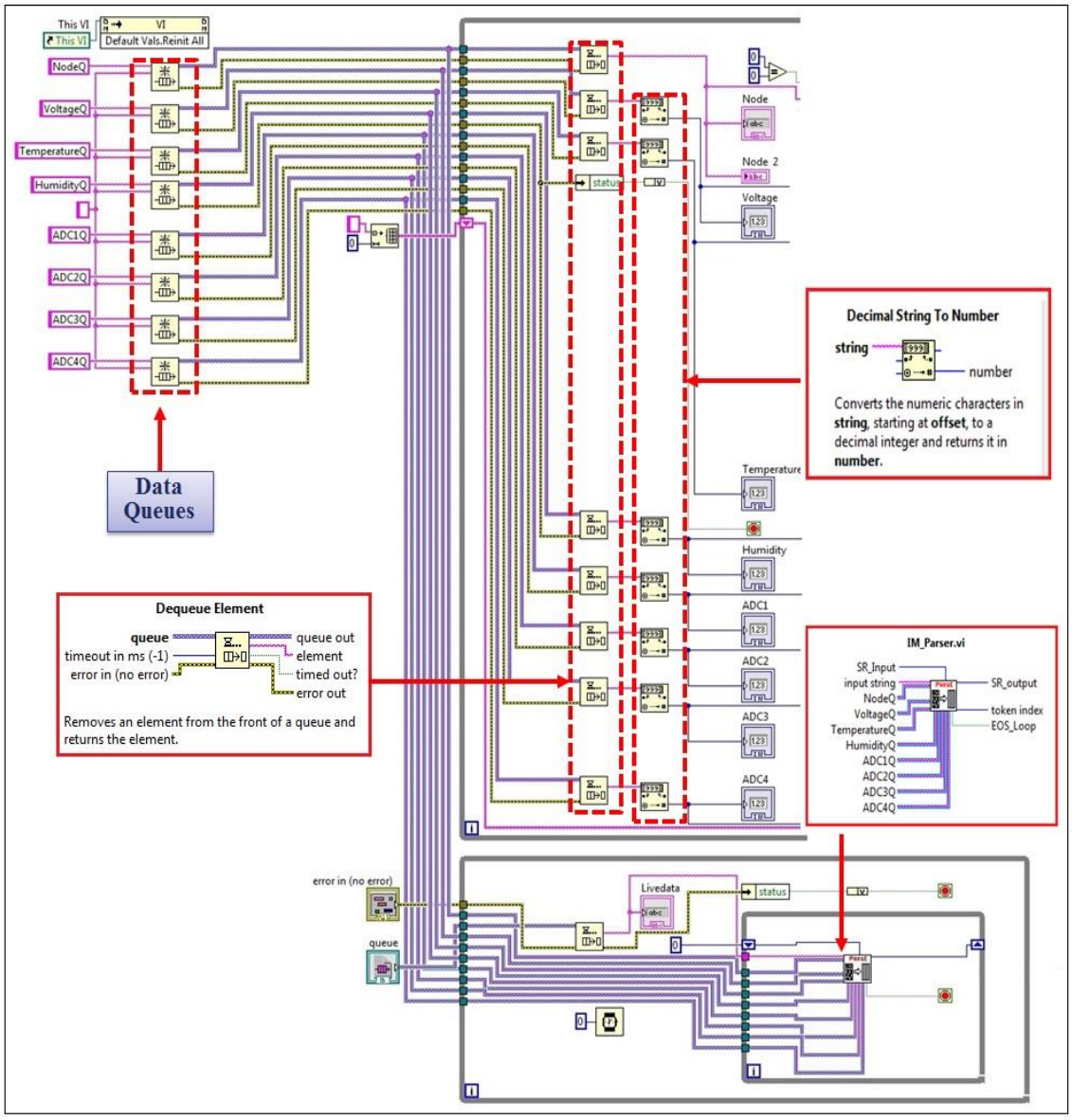


Figure 5.22 LabVIEW implementation of the Sensor Processor component.

empty strings or incomplete strings from the data stream. It has five sub components; Node Selector, Environmental Node, Machine Node, Node Instance Creator and Node Instance Update. Figure 5.22 shows the implementation of the Sensor processor component. The data arrives via data queues from the IM_Parser VI and it gets off loaded using the Dequeue Element function. At this point, the data is in string format so it has to be converted into numerical version using the Decimal String to Number function. It is then passed to the Node Selector sub-component that is implemented as subroutine in this VI.

5.5.2.2.1 Node Selector Component Implementation

The Node Selector sub component is implemented using a Case Structure subroutine in the IM_Sensor_Processor VI. Figure 5.23 shows its implementation. A 1D array is used to store node addresses. If a node address

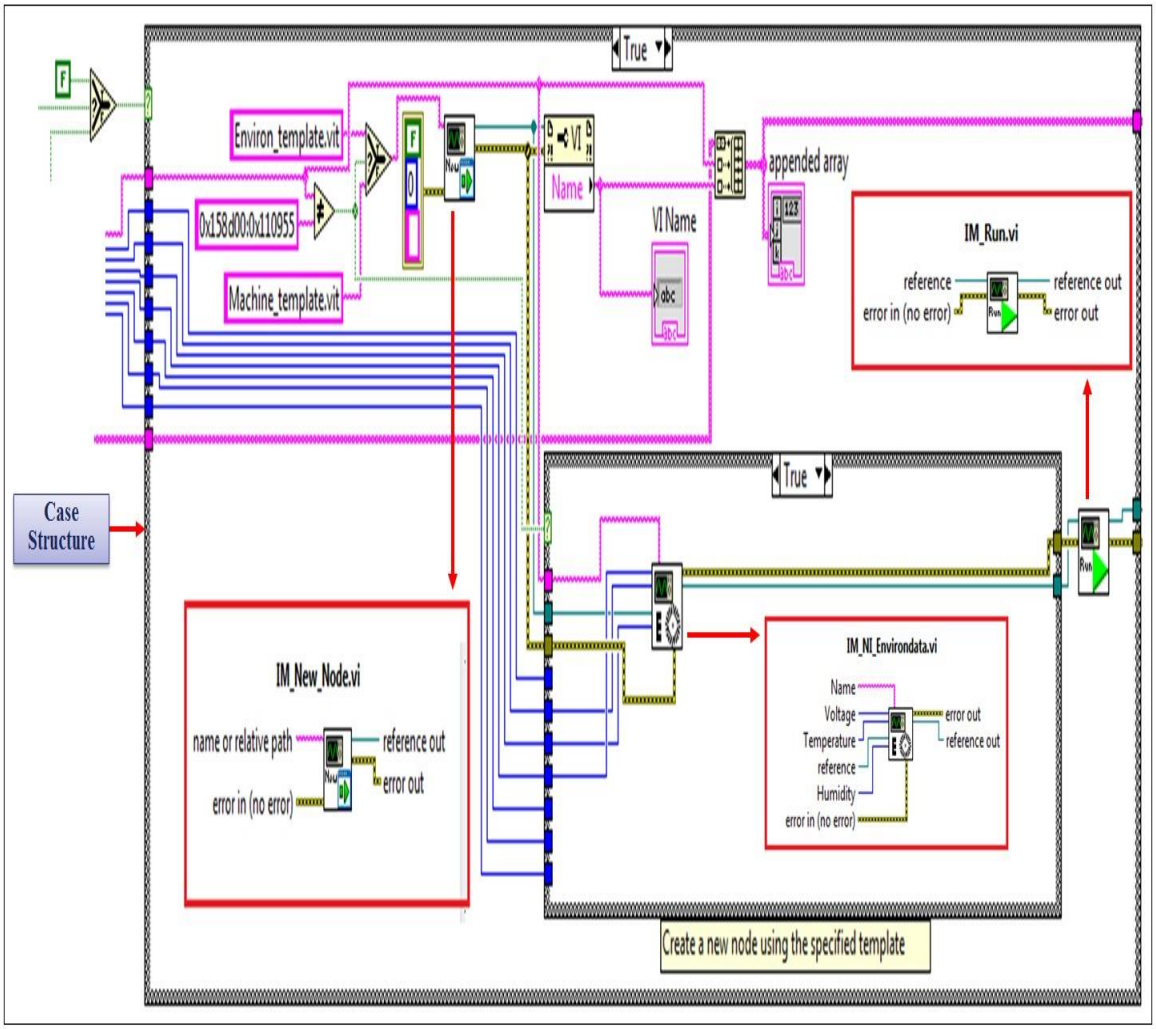


Figure 5.23 LabVIEW implementation of the Node Selector component.

is new it is added to the array and the node selector then uses the case structure to create a new node using a template. It then sends the data to the Node Instance Creator module implemented using the IM_New_Node VI and IM_Run VI. Otherwise, it sends the data to the Node Update component that is implemented using the IM_Update_Node VI. It also sends the data to the IM_NI_Environdata VI and IM_NI_Machinedata VI depending upon the type of node.

5.5.2.2.2 Node Instance Creator Component Implementation

The Node Instance Creator component is implemented in the IM_Node_New VI and IM_Run VI. This component will only receive data if a new node has been detected. It prepares to start a new dynamic instance of an application. It also receives the relative path for the location of the template files to be used for thenew dynamic application. Figure 5.24 shows its implantation code of the IM_New_Node VI. The IM_Run VI shown in Figure 5.25 is invoked to start the new dynamic application once all the relevant information has been received and initialised. It uses the FP.Open function to open a new LabVIEW front panel window at runtime and populates it with the relevant data using the Invoke Node function.

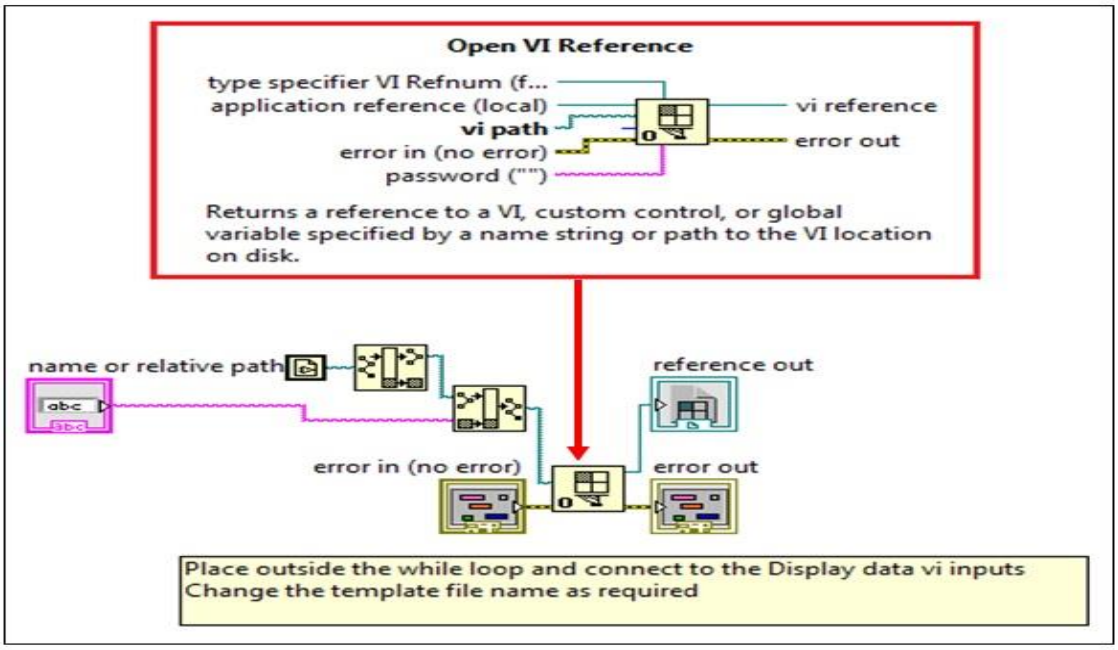


Figure 5.24 LabVIEW implementation of the Node Instance Creator component.

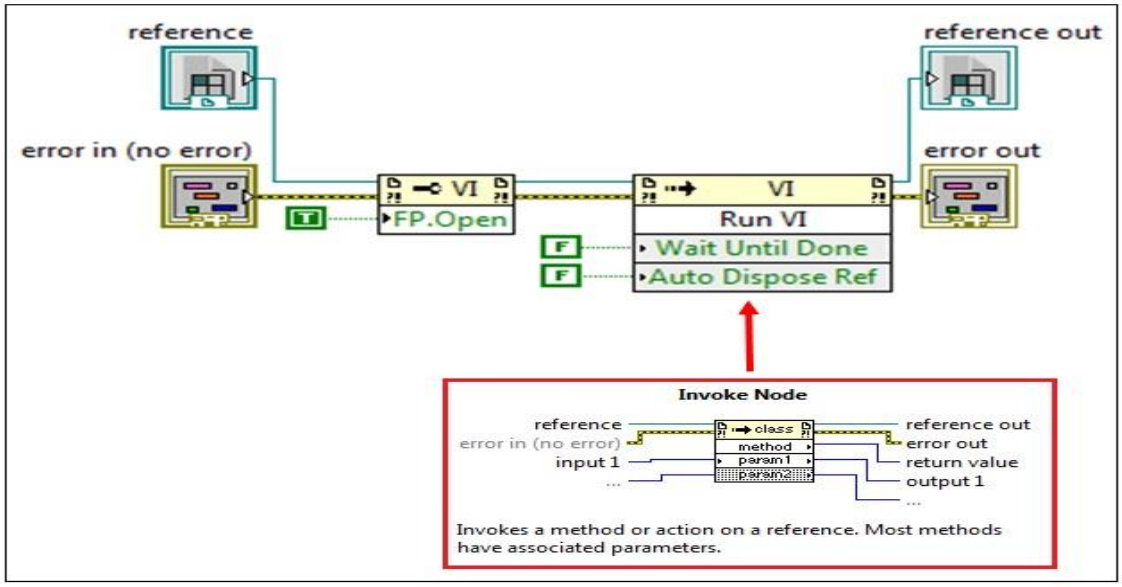


Figure 5.25 LabVIEW implementation of the IM_Run VI used in Node Instance Creator component.

5.5.2.2.3 Node Instance Update Component Implementation

The Node Instance Update component is implemented in the IM_Node_Update VI. It receives data for the nodes that are already open and is responsible for updating these nodes with relevant data using the Invoke Node function and the application reference. Figure 5.26 shows the implementation code for the Node Instance Update component,

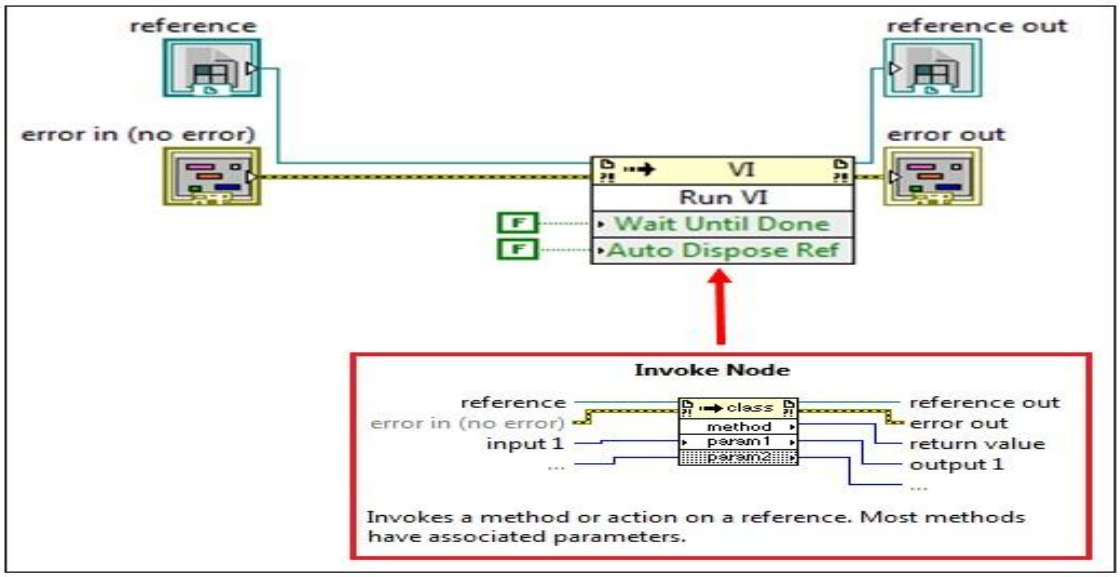


Figure 5.26 LabVIEW implementation of the Node Instance Update component..

5.5.2.2.4 Environmental Node Component Implementation

The Environmental Node component is selected in a case structure in the IM_Sensor_Processor VI illustrated in Figure 5.27. As the address of the the

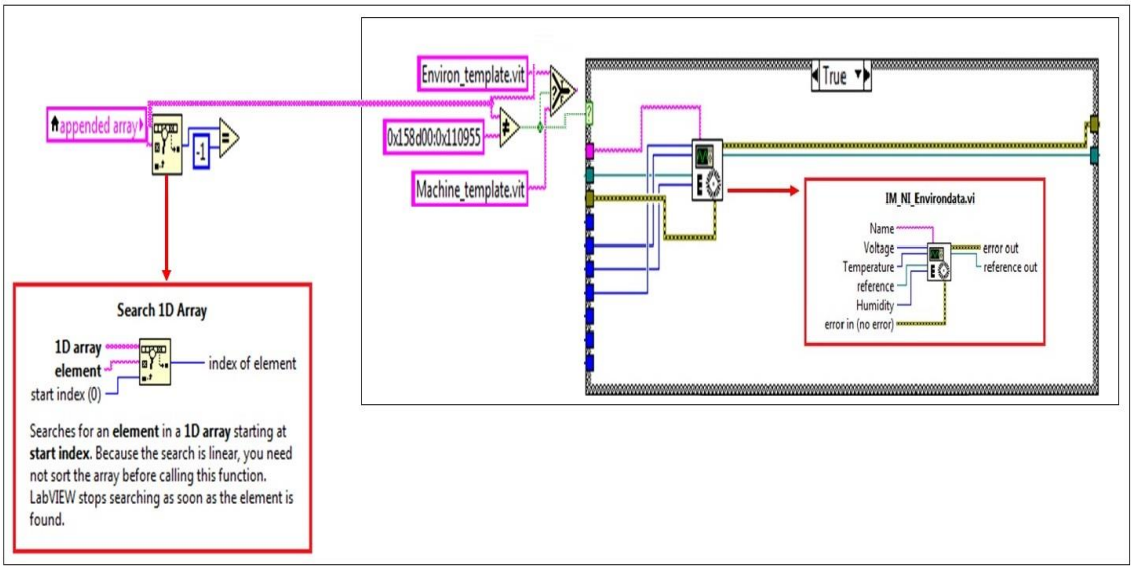


Figure 5.27 LabVIEW code for selecting an Environmental node.

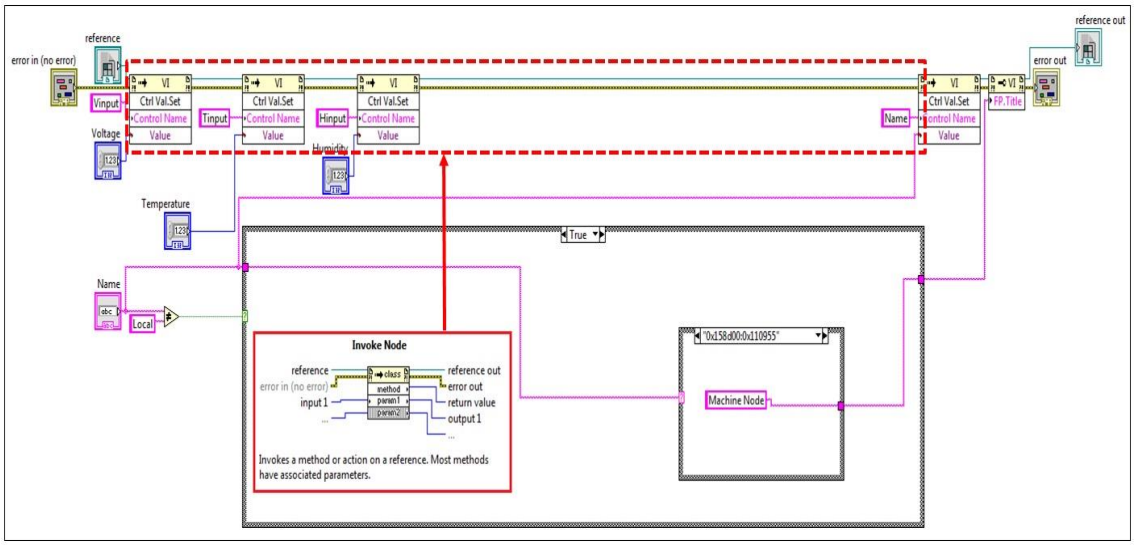


Figure 5.28 LabVIEW implementation of the Environment Node component.

machine node and environmental nodes are known beforehand, the incoming node address gets compared to the known node addresses and based on the result the Environmental node will get selected. The Environment Node component is implemented in the IM_NI_Environdata VI and the implementation code is shown in Figure 5.28. In this code using the Invoke Node function, the data is sent either to the Node Instance Creator component or to the Node Instance Update component depending upon the reference provided.

5.5.2.2.5 Machine Node Component Implementation

The implementation of the Machine Node component is very similar to the Environment Node component. It gets selected in a Case structure in the

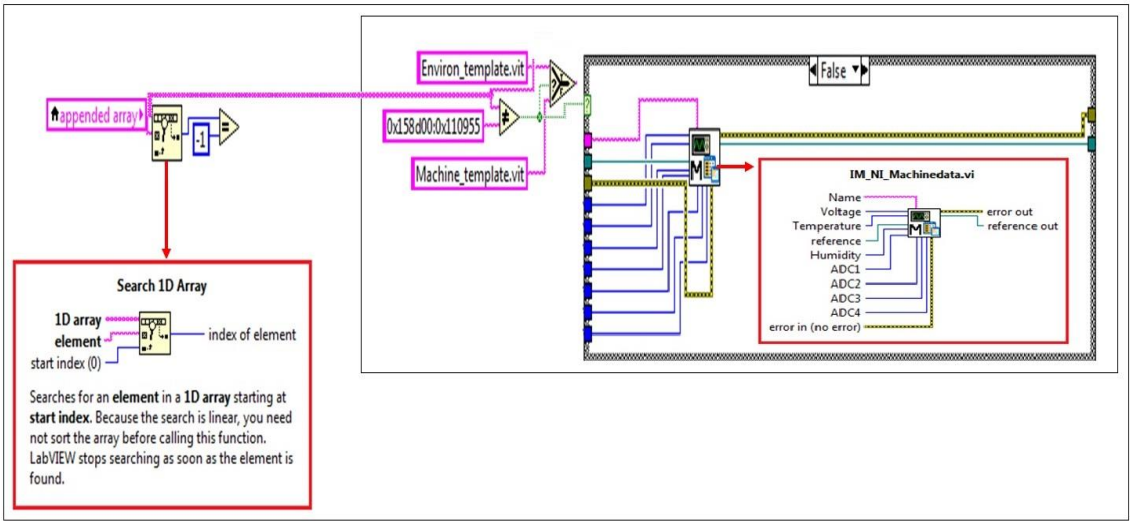


Figure 5.29 LabVIEW code for selecting a Machine node.

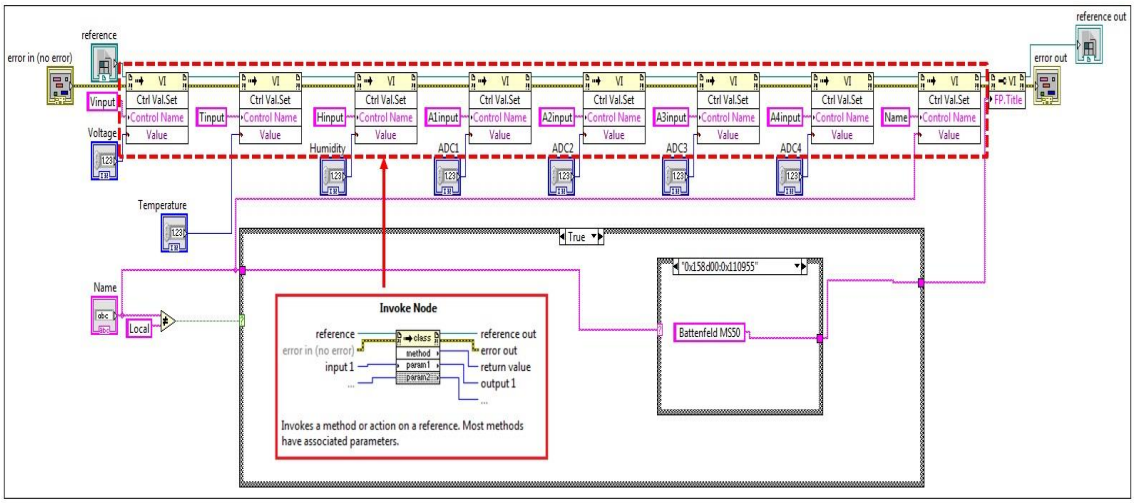


Figure 5.30 LabVIEW implementation of the Machine Node component.

IM_Sensor_Processor VI as illustrated in Figure 5.29. As the address of the machine node and environmental nodes are known beforehand, the incoming node address gets compared to the known node addresses and based on the result the Machine node will get selected. The Machine Node component is implemented in the IM_NI_Machinedata VI and the implementation code is shown in Figure 5.30. In this code using the Invoke Node function, the data is sent either to the Node Instance Creator component or to the Node Instance Update component depending upon the reference provided.

5.5.3 Dynamic Sensor Node Templates layer Implementation

The Dynamic Sensor Node Instances layer has been implemented using LabVIEW's Virtual Instrument Templates (VIT). VI Templates can be considered as blue prints or models, used as starting points to create a new instance (copy) of that template (National Instruments, 2011a). The Dynamic Node Instances layer creates dynamic applications based on the custom templates created for the sensor nodes. The frontend GUIs for the templates are implemented in this layer whereas the functionality code for these templates is implemented in the Dynamic Node Instances layer in section 5.5.4.

5.5.3.1 *Environmental Node Templates*

The data from the environmental node is used by the Environmental Node Templates. The templates GUI is the main place of interaction for the LVWSM application users. Two versions of the Environmental Node template have been created; graphical version and numerical indicator version. The graphical versions GUI was implemented in the front panel of the Environ_template.VIT file whereas the numerical indicator version was implemented in the EnvironNum_template.VIT file. The complete code for these templates can be found in Appendix D.3. More detailed explanation of how the different parts of this this code were implemented can be found in the Dynamic Node Instances layer implementation in section 5.5.4

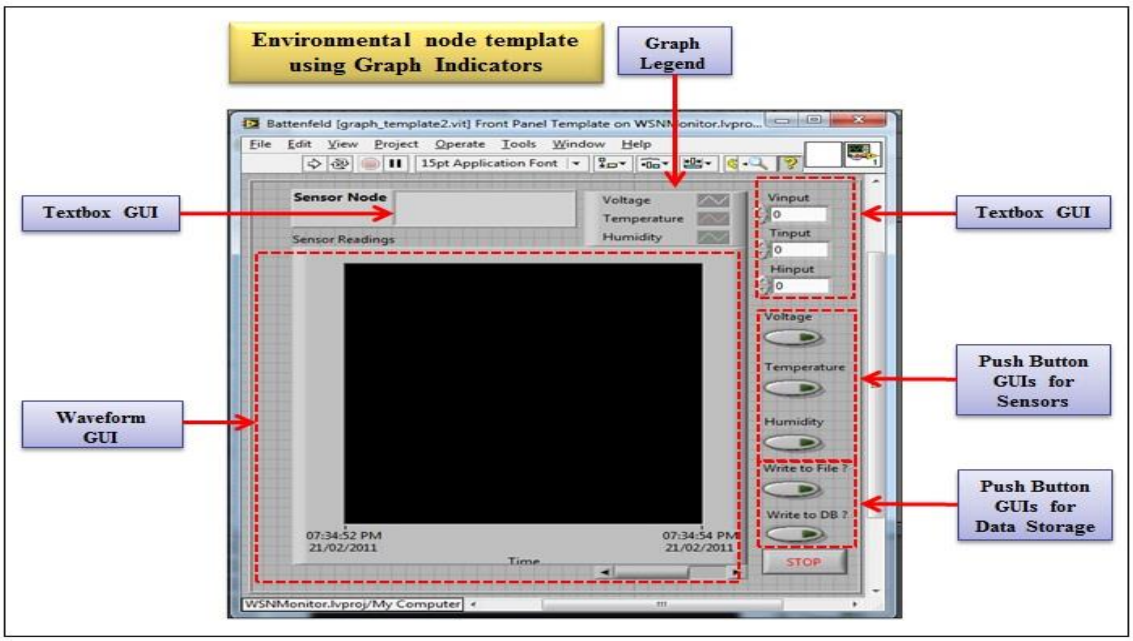


Figure 5.31 Environmental Node template using Graph Indicator.

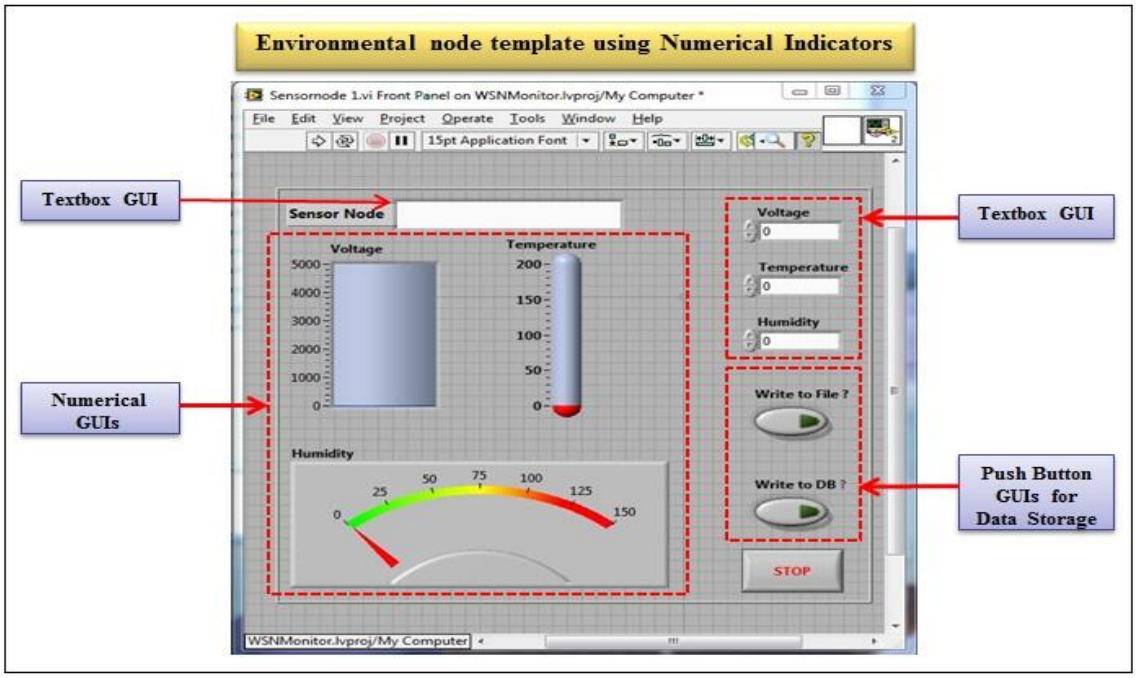


Figure 5.32 Environmental Node template using Numerical Indicators.

Figure 5.31 illustrates the graphical version, which displays the waveform GUI in the centre of the template. This waveform GUI plots the selected sensor readings with the sensor value displayed on the Y-axis and the time displayed on the X-axis. The sensors are selected using push buttons shown on the right. Two push buttons are also used to give the user the choice of logging the data to a file or a relational database. Textbox GUIs are used to display the sensor values as well as sensor node name and address. Figure 5.32 shows the numerical version of the environmental node. This template uses LabVIEW's own Numerical indicator GUIs. A gauge is used to display the humidity; a thermometer is used to display the temperature, and a battery level indicator is used to display the voltage. Textboxes are used to display the sensor readings along with sensor node name and address. Two push buttons are used to give the user the choice of logging the data to a file or a relational database.

5.5.3.2 *Machine Node Template*

The Machine Node Template uses the data from the machine node. This templates GUI is the main place of interaction for the LVWSM application users. Only a graphical version was implemented for the Machine Node template. The GUI for this template was implemented in the front panel of the Machine_template.VIT file. The complete code for this template can be found in Appendix D.3. More detailed explanation of how the different parts of this this

code were implemented can be found in the Dynamic Node Instances layer implementation in section 5.5.4. Figure 5.33 shows the template for the machine sensor node. This template has all the GUIs as the environmental nodes graphical template with additional push buttons to display the machine cavity pressure, nozzle pressure, piston velocity, and piston displacement process variables.

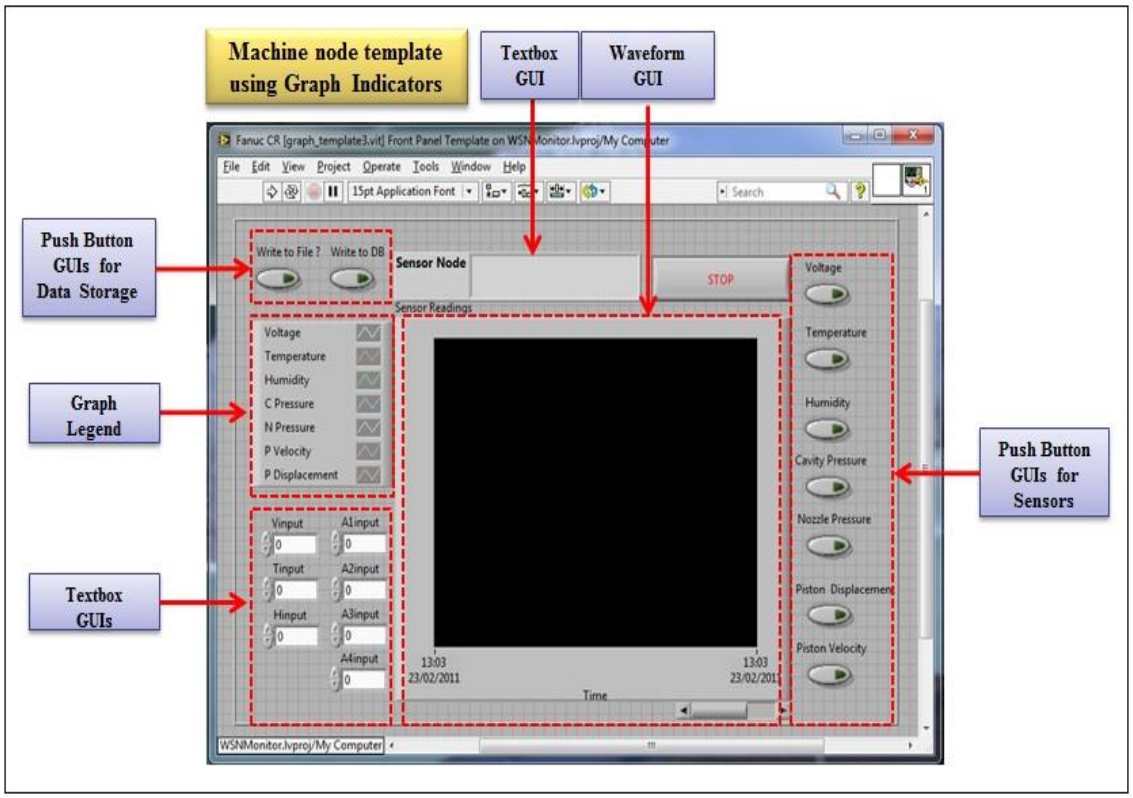


Figure 5.33 Machine Node template using Graph Indicator.

5.5.4 Dynamic Node Instances Implementation

The Dynamic Node Instances layer implements the code for the node templates described in the previous section. For each type of sensor node template the underlying code was similar. The only difference between the Environmental node template and the Machine node template implementation was that the Machine node template had extra inputs for the machine sensors and a calibration VI was used to calibrate the machine sensors. The complete implementation code for both types of nodes can be found in the sub sections of Appendix D.2. The implementation code for each component of this layer is described in more details next.

5.5.4.1 *Write to File Component Implementation*

The Write to File component is used to log node data to a file. This component was implemented in the node template code as a subroutine and uses LabVIEW's File I/O functions to implement the logging functionality. From the implementation code shown in Figure 5.34, the Open/Create/Replace File function is used to open a file for reading and writing. The Set File Position function is used to append the data to the end of a file if it already exists. A Concatenate String function is used to create and format the column headings for each variable of the sensor node. The data is written using the Write to Binary File function and Format into File function is used to format the data so that it is written under the right heading. The Get Date/Time function is used to write the current date and time stamp into the file. The data is only logged into a file when the Write to File bush button is activated on the front panel.

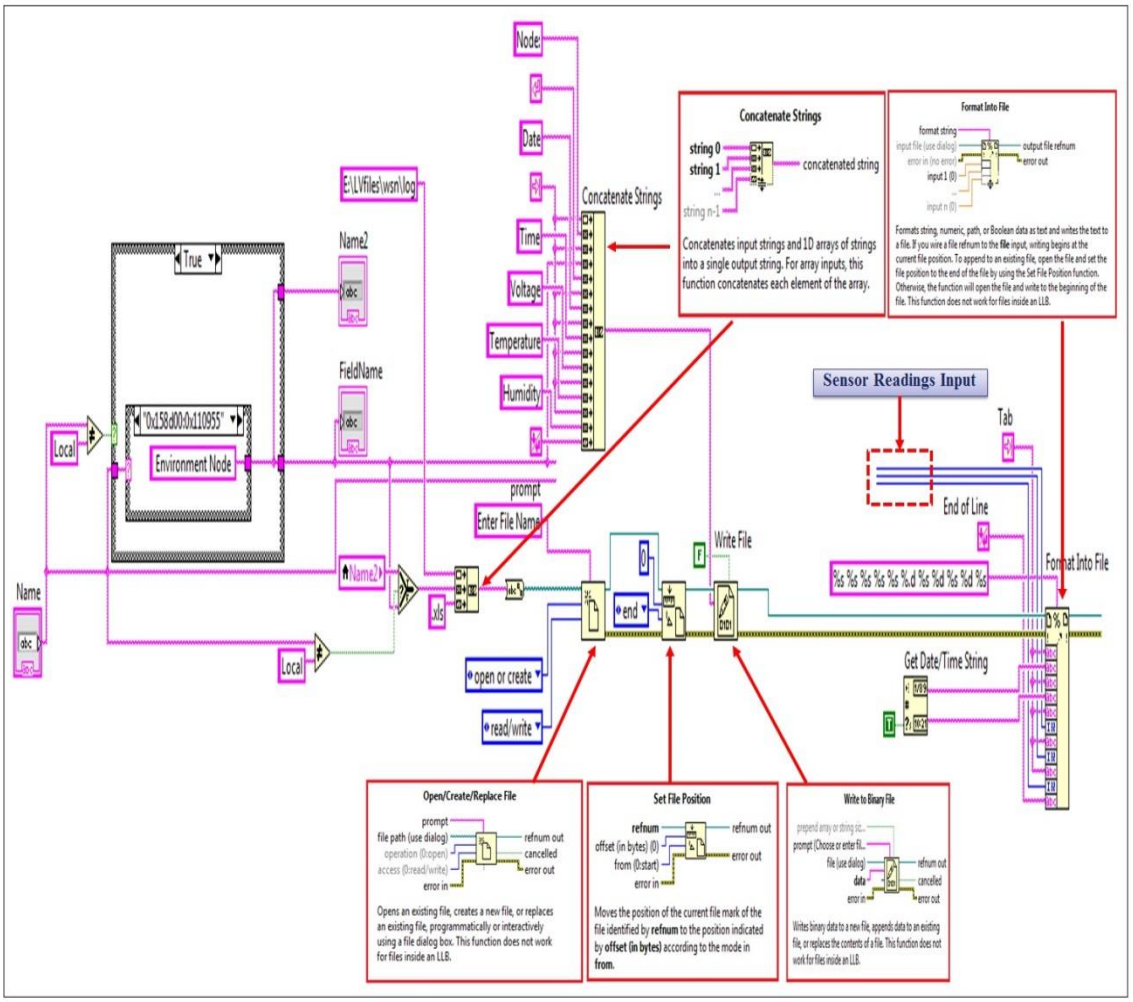


Figure 5.34 LabVIEW implementation of the Write to File component.

5.5.4.2 *Write to DB Component Implementation*

The Write to DB Component is used to log the data into a relational database. This component was implemented in the node template as a subroutine and uses LabVIEW Database Connectivity functions to implement the database logging functionality. Figure 5.35 shows its implementation in which the DB Tools Open Connection function is used to connect to a database using either the Universal Data Link (UDL) file or the Open Database Connectivity (ODBC) interface. The DB Tools Insert Data function is used to create a table with the specified name and the columns with the provided column names. If the table and columns do not exist, it will create them automatically. It then inserts the node data into the relevant columns. The data is only logged if the Write to DB push button on the front panel is activated.

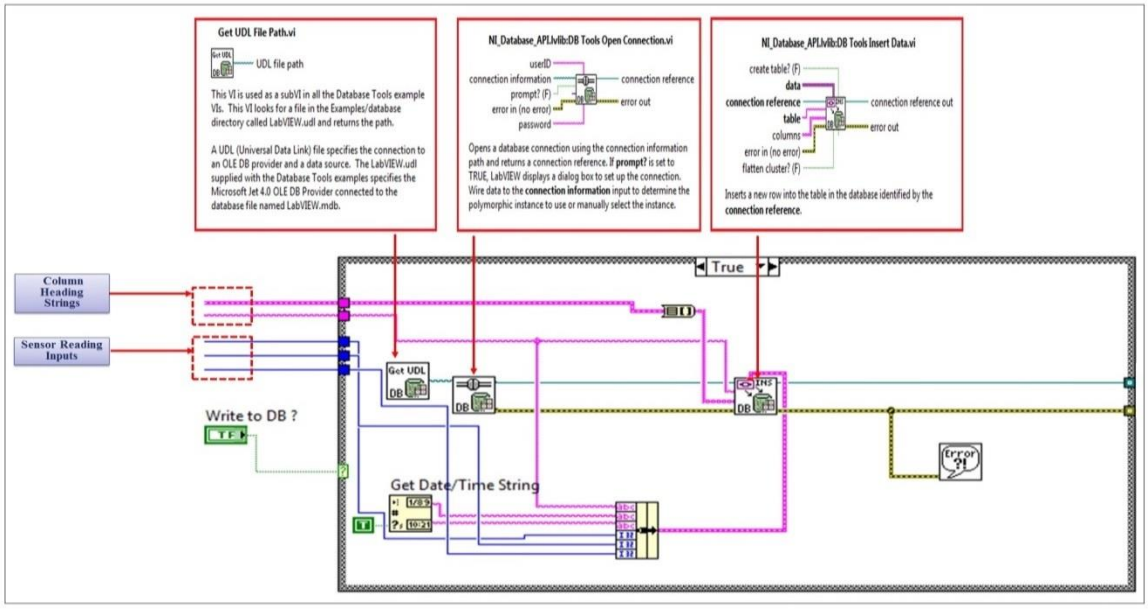


Figure 5.35 LabVIEW implementation of the Write to DB component.

5.5.4.3 *Create Scale Component Implementation*

The Create Scale Component is used to create and dynamically change the XY axis scales when the view is changed between the different sensors. This component is also implemented as a subroutine and Figure 5.36 show its implementation code. The subroutine is implemented in a separate while loop with an Event Structure in the loop. The Event Structure waits for any events to take place such as clicking of a button and then executes the relevant code for each case.

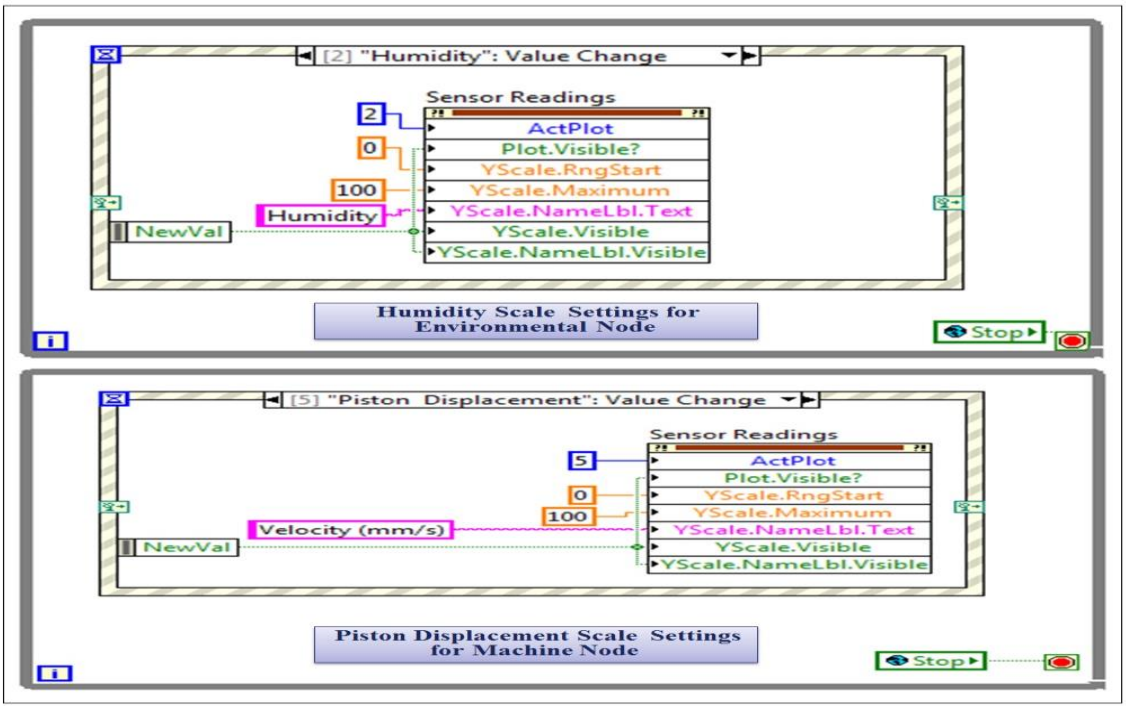


Figure 5.36 LabVIEW implementation of the Create Scale component.

5.5.4.4 *Display Graph Component Implementation*

The Display Graph component is implemented in a subroutine and is responsible for plotting the live sensor data readings. Figure 5.37 shows the implementation of this component using LabVIEW's Waveform Charts function.

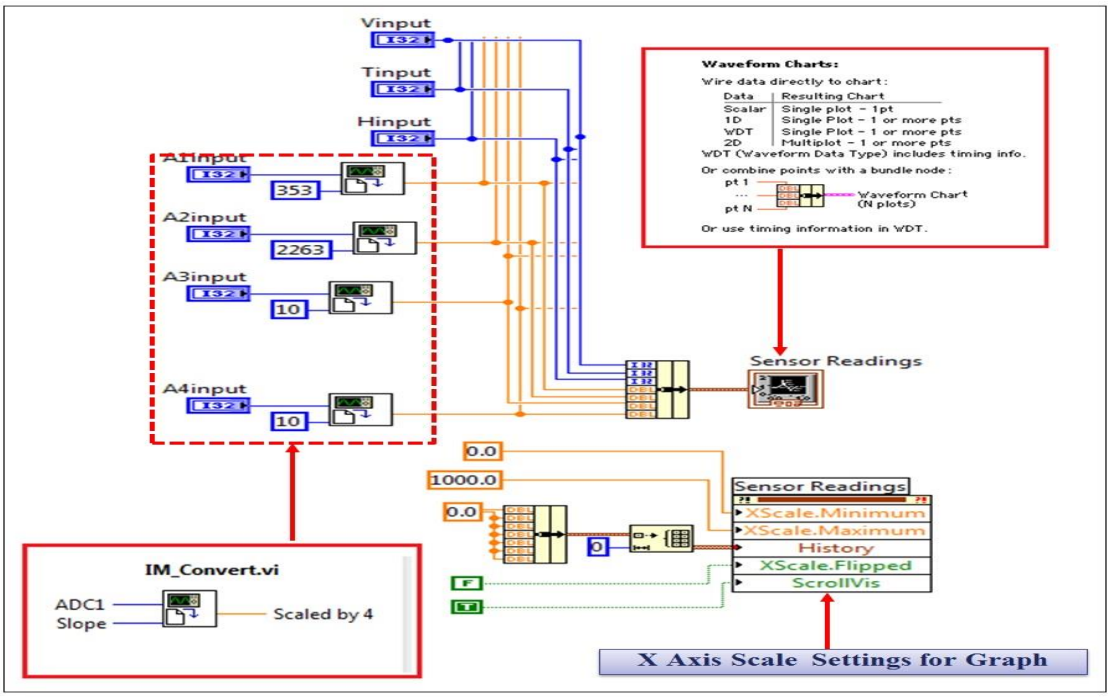


Figure 5.37 LabVIEW implementation of the Display Graph component.

All the node data is bundled together and then fed into the Waveform Chart, which plots each input with time. The graph is updated every second. The inputs from the ADC channels are fed into a simple VI called IM_Convert, which was created to convert the ADC readings to analogue form and then calibrating them using the data provided by the manufacturers.

5.5.4.5 *Sensor Web Data Component Implementation*

The Sensor Web Data component makes the node data available for the Sensor Web layer to be published as a Web Service and is implemented in two VI files (one for each node); IM_EnvironWebData VI and IM_MachineWebData VI. These VIs are called from within the node templates main loop and make the node data available by feeding each sensor reading into a shared variable. Figure 5.38 shows the implementation of this component for both the environmental and machine nodes.

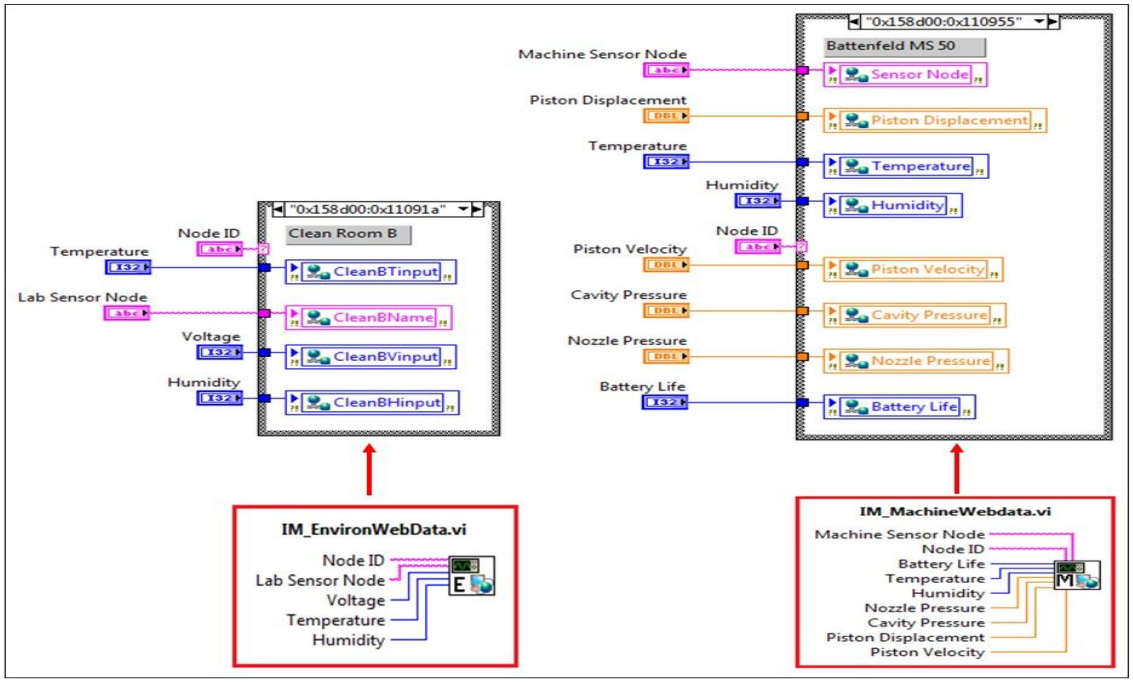


Figure 5.38 LabVIEW implementation of the Sensor Web Data component.

5.5.5 Sensor Web layer Implementation

The Sensor Web layer has four components; Sensor Web Service, Create WS, Deploy WS, and URL Mappings. It is responsible for converting the node data into Restful Web Services. The Create WS, Deploy WS and URL Mapping components are user implemented through the build specification interface in

the project explorer. These components are used when the Web Service templates are created at the beginning with the relevant URL Mappings and deployed on the local LabVIEW Web Server. Once the Web services are deployed on the server, the Sensor Web Service Component can invoke them.

5.5.5.1 *Sensor Web Service Component Implementation*

The Sensor Web Service Component is the Web method responsible for organising the Web Services data as well as handling requests from the Web clients through the URL Mappings component. It is implemented in two VI files (one for each node); IM_EnvironWebservice VI and IM_MachineWebservice VI. In the implementation code shown in Figure 5.39 the node data request from the Web client are received as an input to a case structure. Depending upon the request received the relevant case is selected. For example if the Web client requests data for the node with address 0x110955, then the relevant case structure requests the data from the Sensor Web Data component. The data is received through the shared variables and is returned back to the requesting client.

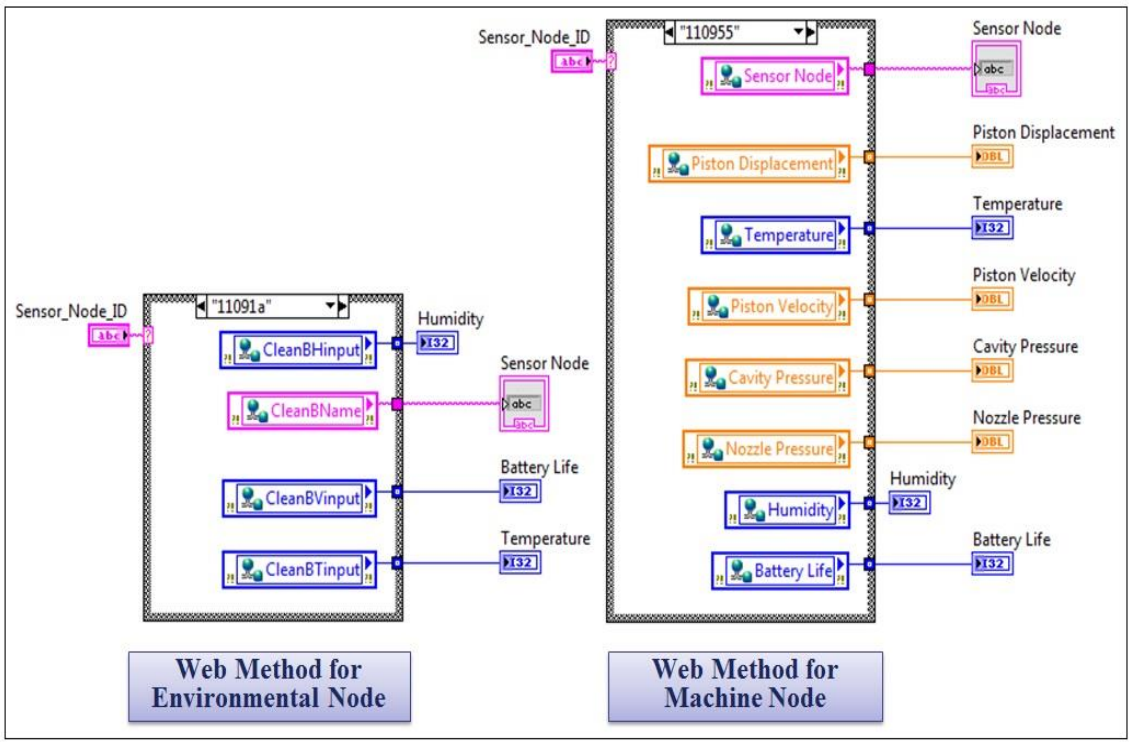


Figure 5.39 LabVIEW implementation of the Sensor Web Service component.

5.6 LVWSM System Testing and Evaluation

This section focuses on the testing and evaluation of the LVWSM µIM process monitoring prototype system based on the proposed SOA based architecture presented and implemented in the previous sections of this chapter. The individual layers of the architecture are tested before integrating them together as a complete prototype system. The prototype system as a whole is then tested and evaluated in terms of its GUIs, functionality, data collection and web services. First, the system was tested for its functionality of data logging using files and relational databases. In the first test, the system was used in the calibration of the nodes on-board sensors. The second test involved the monitoring of the two clean rooms in the polymer MNT lab. In the third test the environment in this case the cleanroom and the Battenfeld Microsystem 50 µIM machine was monitored using the machine interface board. In the final test, the Web Services created using this system were utilised for integrating with other web applications. This was demonstrated by in two scenarios. First using the LV Web UI builder application (National Instruments, 2011b) and then using the Jackbe Presto mashup application (Jackbe Corporation, 2011).

5.6.1 Data Monitor layer Test

The Data Monitor layer was tested on its own to verify that the data captured by the serial port is being transferred into the buffer. To carry out this test, the Highlight Execution button on the block diagram (a debugging tool that animates the execution of the VI) was used in conjunction with Single Step debugging. The frontend GUI was used with indicators to display the captured data per iteration. The test was carried out in two steps: First, the Coordinator node attached to the PC was switched on to start the WSN; Second the End Device sensor nodes were switched on next which joined the network and started to transmit data. To verify that the data was being captured, the GUI shown in Figure 5.20 was used to view the data.

5.6.1.1 *Data Monitor Test Results*

The results from this test are shown in Figure 5.40 in which the output on the left shows the data captured when the Coordinator node starts the network whereas the output on the right shows the data transmitted by the machine

node which has joined the network. From these results it can be seen that the data being sent by the WSN is being captured as expected.

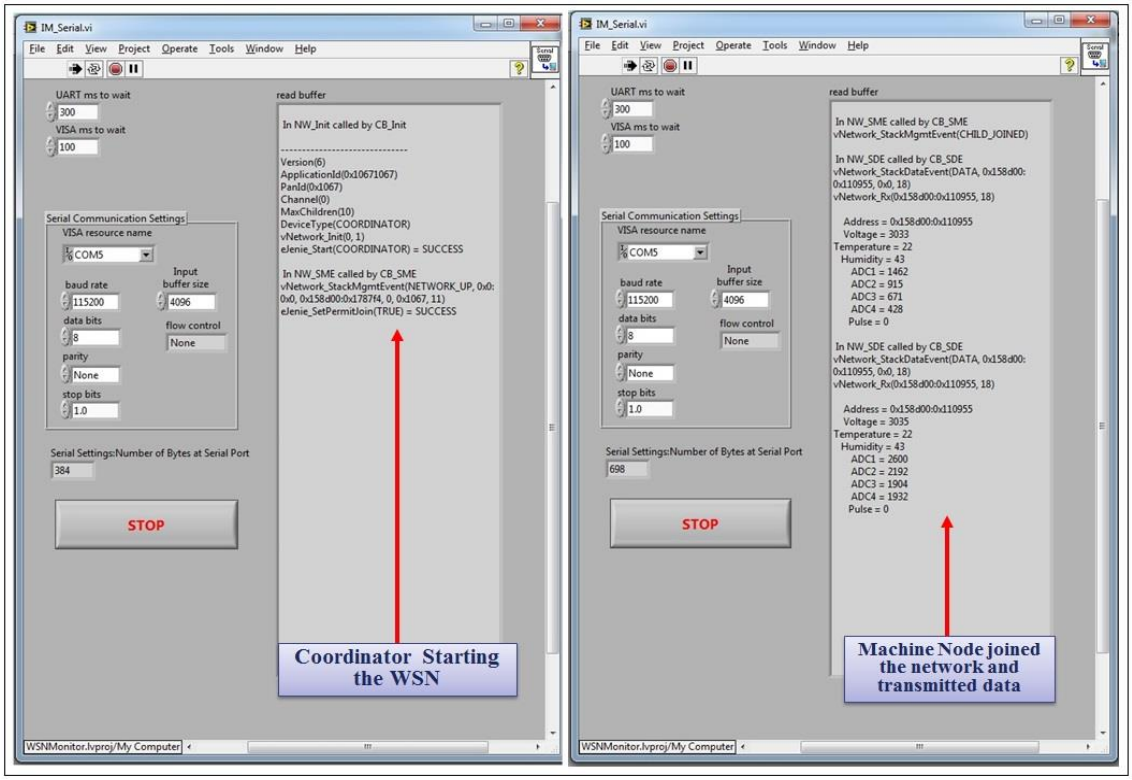


Figure 5.40 Data Monitor test results.

5.6.2 Data Processing layer Test

The Data Processing layer was tested using the same methods as in the previous test. In this test, the main components of this layer were tested in the following sequence:

- First, the Data Parser component was tested to verify that the String Tokens were being used correctly to separates the key data from the data stream.
- Secondly the Sensor Processor component was tested for the following:
  - To verify that the correct separated live data was being displayed.
  - To verify that the right node type is being identified using this data.
  - To verify that the right template is being selected using this data.

The test data used was sent from the machine sensor node with the address 0x110955.

5.6.2.1 *Data Parser Test Result*

Figure 5.41 shows the used data tokens on the left and the live data input string being input to the parser component in the middle. The results from the application of these data tokens to the input string show that the data being input to the parser component is being separated as expected. The separated key data is displayed in each token indicator and is from the latest sensor node readings. From this test it can be verified that the separated key data is from the latest sensor node readings and that the parser component is working as expected.

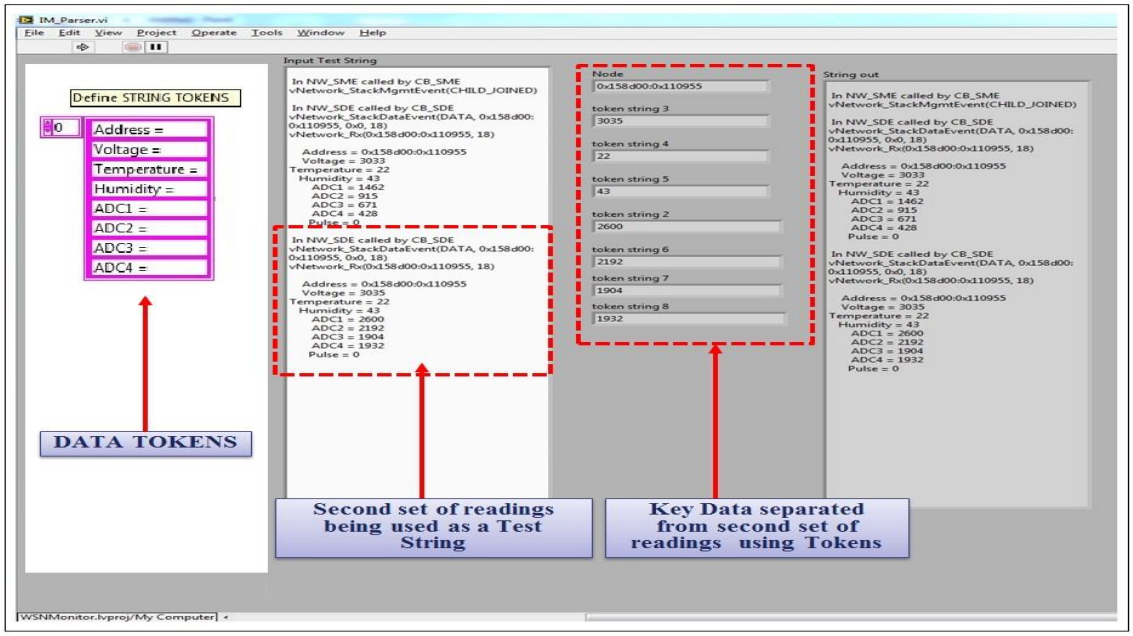


Figure 5.41 Data Parser test results.

5.6.2.2 *Sensor Processor Test Result*

Figure 5.42 shows the key data separated by the Data Parser component being used by the Sensor Processor component to select the relevant node type. It also selects the relevant template to be used for the new instance of the application to be started. From this test result the following points could be verified:

- The separated data being displayed was the same as the one provided by the Data parser component.
- The data was identified to be from a machine node based on the node address.
- The Machine_template VI was selected based on the node address.

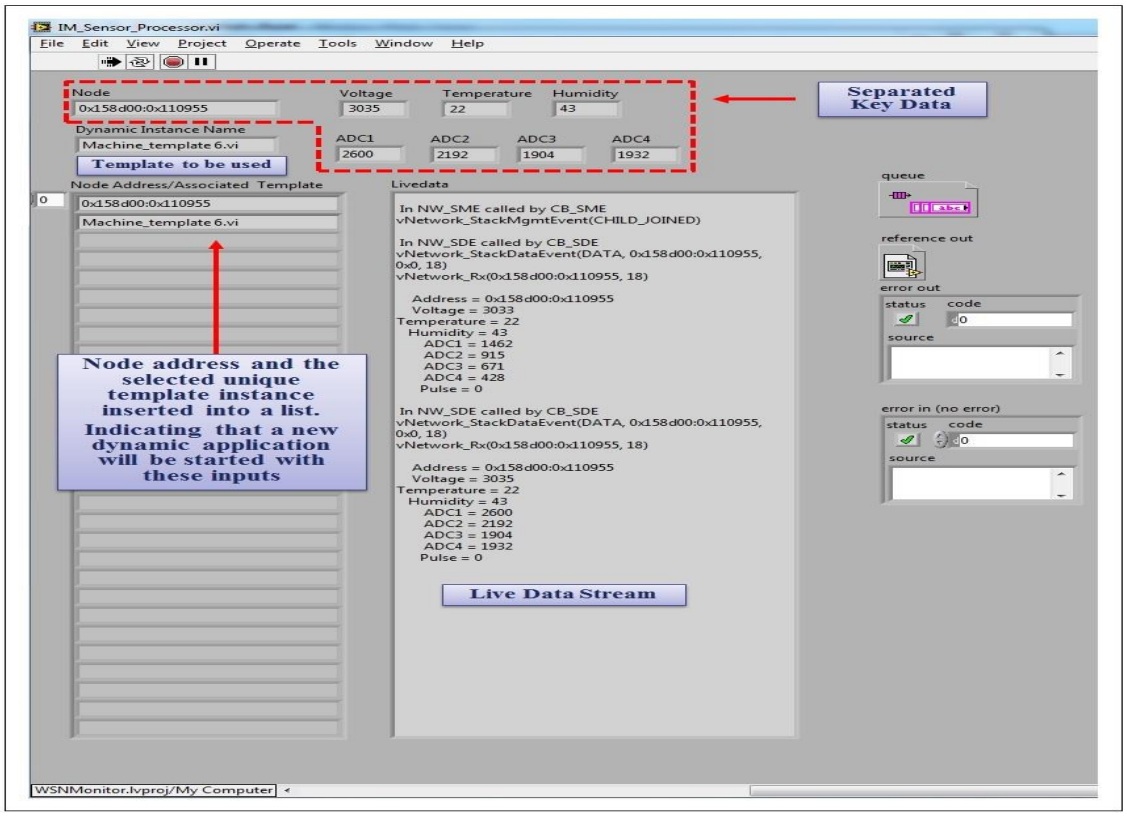


Figure 5.42 Sensor Processor test results.

5.6.3 **Dynamic Node Instances and Dynamic Sensor Node Templates layer Test**

In order to test that the Dynamic Node Instances layer was functioning correctly by creating the relevant dynamic node instances; node templates provided by the Dynamic Sensor Node Templates layer have to be used. Therefore, it made sense that both these layers are tested together. The same method used in the previous tests was utilised to carry out this test. In this test, the data provided by the Data Processing layer is used to start a dynamic instance of an application based on the provided template. Therefore, the following points need to be verified:

- The correct instance of an application is being started using the provided template.
- The correct GUIs are displayed in the right place.
- The application is initialised properly with the right initial values.
- The GUIs function as expected
  - ➢ The sensor display function

- The XY Scales display correctly when changing between sensors.
- Sensor readings are plotted correctly in the display graph.
- The Write to File function
- The Write to DB function

This test was carried out using data from the machine sensor node with address 0x110955. The same test was carried out using data from an Environmental Node with address 0x11091a. The test results for the machine node are shown in the following section.

5.6.3.1 *Dynamic Node Instances Test Results*

Figure 5.43 shows the result in which a new dynamic instance of the machine node is created using the data from the machine sensor node. The following points can be verified from the results in this figure:

- The data values being displayed are the correct values and are the first set of values sent by the machine node. The second sets of values have yet to be updated as the execution was paused for this test.
- All the GUIs are being displayed in the right places.
- The sensor push button GUIs functioned correctly and the resultant graph is being displayed in the graph display screen.
- The graph displays the correct values.
- The scales for the new selected sensor have updated correctly (this was tested for all the sensor push buttons).
- The correct scale legend was highlighted for the selected sensor.
- The correct node name was being displayed
- Additional information such as the deployed web services could also be verified.

Figure 5.44 shows the data logged when the Write to File button was pressed. From this result it can be seen that the data was logged into an excel file named with the same name as the machine node. The data logged in the excel file had the correct column heading names and time stamp. The data logged into these columns was the same data displayed in the frontend GUI. The correct node name was logged into the file at the top of the file.

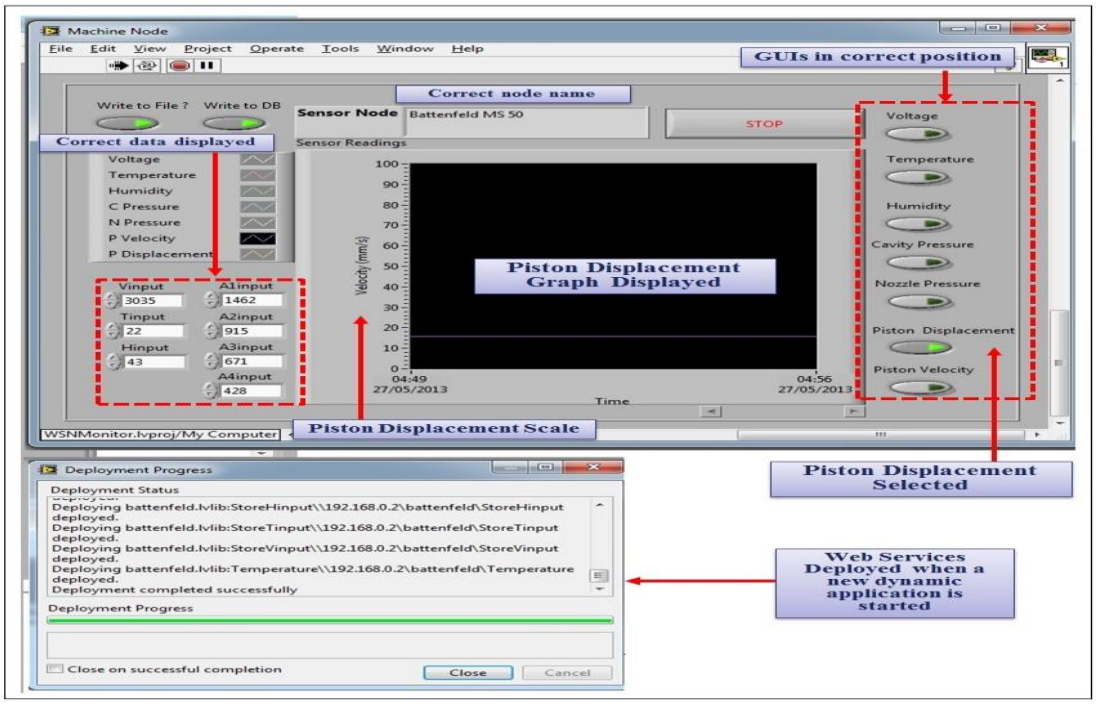
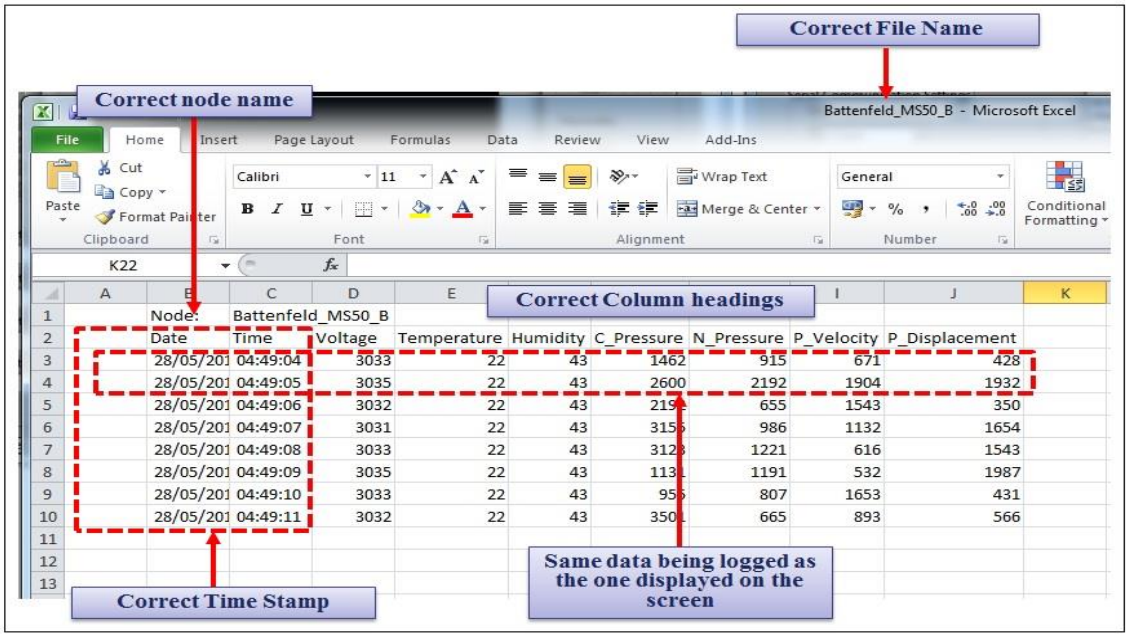


Figure 5.43 Dynamic Node Instances test results.

Figure 5.45 shows the data logged when the Write to DB button was pressed. From this result, it can be seen that the data was logged into an Access database that has a table named with the same name as the machine node. The data logged in the Access database had the correct column heading names and time stamp. The data logged into these columns was the same data displayed in the frontend GUI. The correct node name was displayed at the top of the table.



	A	B	C	D	E	F	G	H	I	J	K
1		Node:	Battenfeld_MS50_B								
2		Date	Time	Voltage	Temperature	Humidity	C_Pressure	N_Pressure	P_Velocity	P_Displacement	
3		28/05/201	04:49:04	3033	22	43	1462	915	671	428	
4		28/05/201	04:49:05	3035	22	43	2600	2192	1904	1932	
5		28/05/201	04:49:06	3032	22	43	219	655	1543	350	
6		28/05/201	04:49:07	3031	22	43	3156	986	1132	1654	
7		28/05/201	04:49:08	3033	22	43	3123	1221	616	1543	
8		28/05/201	04:49:09	3035	22	43	1131	1191	532	1987	
9		28/05/201	04:49:10	3033	22	43	956	807	1653	431	
10		28/05/201	04:49:11	3032	22	43	3501	665	893	566	
11											
12											
13											

Figure 5.44 Write to File test result.

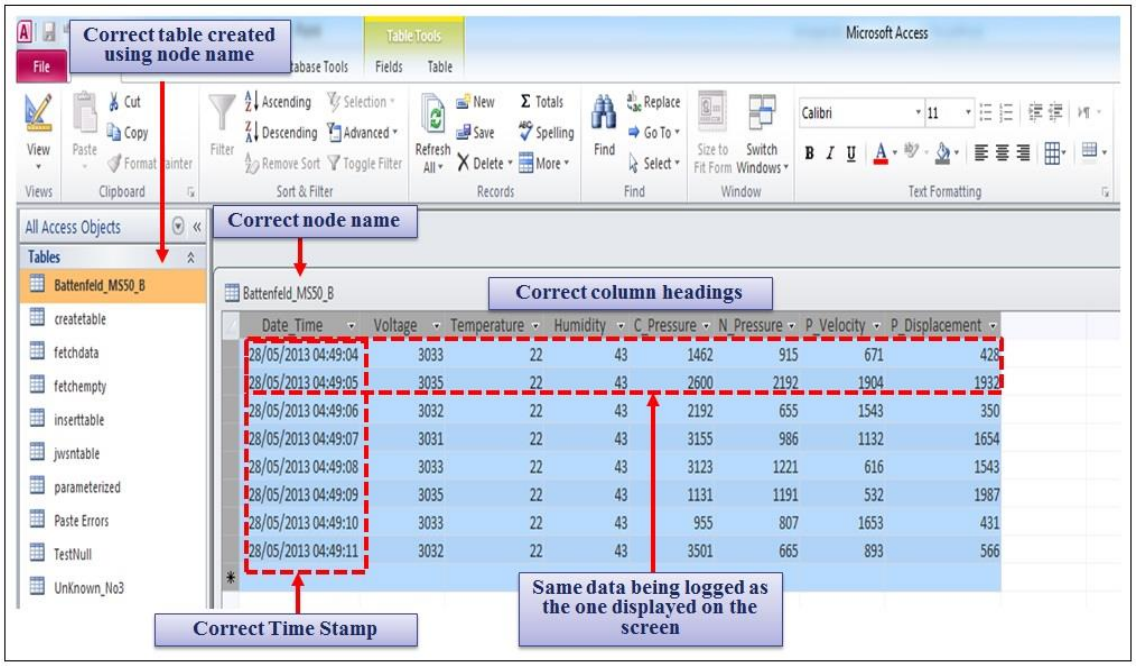


Figure 5.45 Write to DB test result.

5.6.4 Sensor Web layer Test

The Sensor Web layer was tested by invoking the deployed Web Services for the relevant node. In this test, the URL for the machine sensor node was used to invoke the machineWS Web Service deployed on the local LabVIEW Webserver using the port number 8085. The Web Service was invoked from the Mozilla Firefox web browser. The aim of this test was to verify that the deployed Web Services used the correct Web Method to retrieve the relevant node data and that the data was being displayed in XML format.

5.6.4.1 *Sensor Web layer Test Result*

In Figure 5.46 the output on the right shows the result when using the URL http://localhost:8085/machineWS/110955 to invoke the machineWS deployed on the local LabVIEW Web Server. This result verifies that the correct Web service has been invoked and that the data has been retrieved using the underlying Web Method. This result also verifies that the format of the data is in XML. The output on the left shows the result when using the URL http://localhost:8085/MachineWS/11091a to invoke the environWS deployed on the local LabVIEW Web Server. This result also verifies that the result is in XML and the correct data has been retrieved using the correct Web Method.

Figure 5.46 Sensor Web layer test result.

5.6.5 LVWSM µIM Process Monitoring System Prototype Test

After testing the individual layers, the complete LVWS µIM process monitoring system prototype was built by connecting all the layers together. The LVWSM system as a whole was tested in terms of its functionality, ability to monitor the process by logging the data, and integration with web applications using Web Services. The system was evaluated in four tests

- First, the system was tested for its functionality of data logging in a file format by using it in the calibration process of the nodes on-board sensors.
- The second test involved the monitoring of the two clean rooms in the Polymer MNT Laboratory for continues seven days.
- In the third test, the Battenfeld Microsystem 50 µIM machine was monitored using the machine node connected to the interface board.
- In the final test, the Web Services created using this system were used for integrating with other web applications. This was implemented in two scenarios. First using the LV Web UI builder application and then using the Presto mashup application.

5.6.5.1 *WSN Node Calibration Test using the LVWSM*

The LVWSM system was used in this test to calibrate the sensor nodes on-board temperature and humidity sensors. Although from the datasheets from Jennic the on-board temperature and humidity sensors have been calibrated. In one of the previous experiments, the readings from the on-board sensors on each sensor node were noted to be slightly out. Therefore, the calibration of these sensors was necessary. The Climatic test chamber by Weiss Technik described in Appendix B allows the simulation of climatic conditions by allowing the setting of temperature and humidity. For this project, the climatic oven was used for the calibration of on board humidity and temperature sensors on the sensor nodes. The WKL 100 own sensors have been calibrated by the manufacturer and are known to be accurate. The sensor nodes to be used in the clean room monitoring were placed in the WKL 100 climatic chamber. The climatic oven was set to initial values of 38-Celsius temperature and humidity to 10%. Once the chamber reached these values, the temperature was set to 18 Celsius and Humidity of 40%. As the chamber environment started to change to achieve these values, readings were taken from the sensor nodes on-board temperature and humidity sensors using the LVWSM system. These readings were then compared with the climatic oven temperature and humidity readings. The difference between the climatic oven readings and the sensor node reading was noted and a linear curve fitting was used to calibrate the sensors. See Appendix C for the calibration data and graphs.

5.6.5.1.1 **WSN Node Calibration Test using the LVWSM Results**

The LVWSM system was successfully used in the calibration of on-board node sensors. The data gathered using this system was logged into separate files for each node used. Figure 5.47 shows a snapshot of the gathered data. From this, it can be seen that the data is being logged correctly in the right format. After analysing the data, gathered using this system, a small glitch was detected which was logging incomplete data at some points in the file. The problem was pin pointed to the timing synchronization between the system layers. This issue was resolved by modifying and adding some extra code.

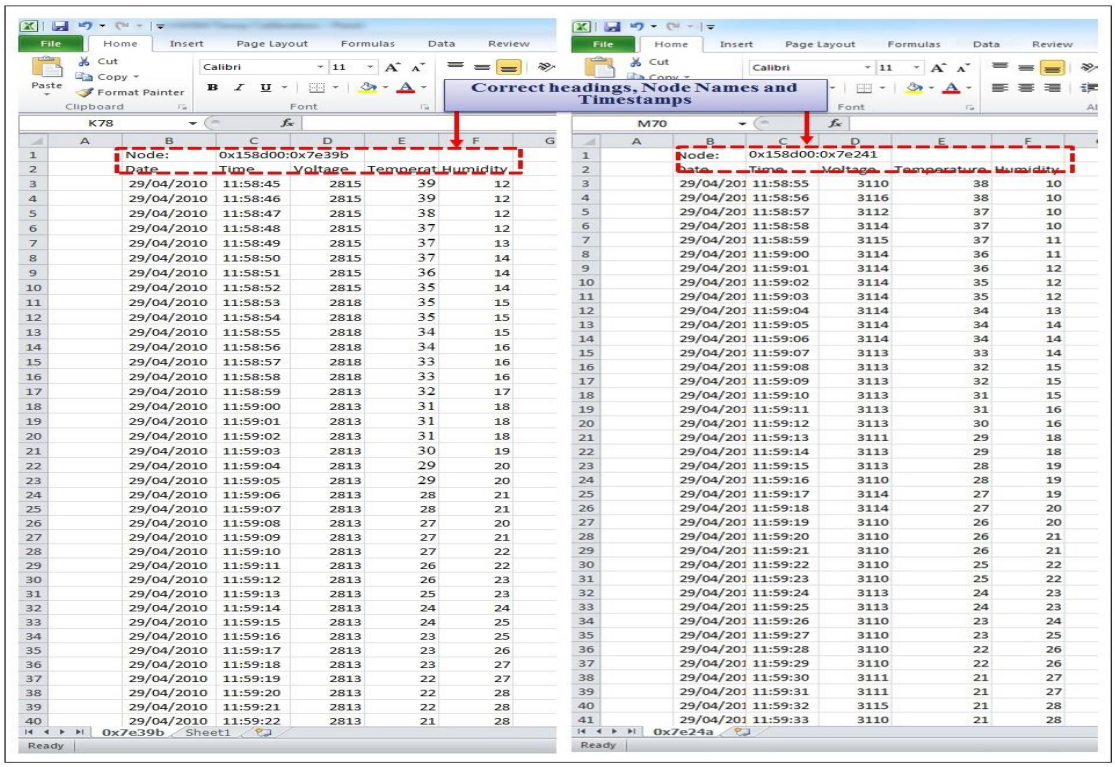


Figure 5.47 WSN Node Calibration test results

5.6.5.2 *Clean Room Monitoring using LVWSM System*

In this experiment, the LVWSM System was setup on one of the laptops in the Polymer MNT Laboratory. The Jennic WSN was deployed with the Coordinator node connected to the laptop with LVWSM System and two sensor nodes placed in each of the clean rooms. Figure 5.48 shows the two clean rooms in the Polymer MNT Laboratory monitored by these sensor nodes and the layout

Figure 5.48 The Polymer MNT Laboratory.

of the polymer MNT Laboratory is shown in Figure 5.8. In this experiment, two tests were carried out using the LVWSM system and the aim of these tests was:

- To monitor the battery life of the nodes used in this experiment.
- To monitor the two clean rooms to detect any patterns changes in the humidity and temperature.
- Test the LVWSM systems capability to monitor for longer periods
- To test the LVWSM systems ability to log large sets of data.

5.6.5.2.1 Clean Room Monitoring using LVWSM System Results

In the first test the sensor nodes firmware was reprogrammed to transmit every five seconds and the nodes were left in the clean room overnight. In the second test, the sensor nodes were left in the clean rooms to monitor the environment for one week using the LVWSM System.

Test one: In this test, the battery life of each sensor node installed in the clean rooms and the router node was monitored and the results are shown in Table 5.1. It was noted that the router node which continuously transmitted data had a reduced battery life time whereas the End Device Environmental nodes that transmitted data every five seconds.

Node Name and Type	Date	Time	Power Source	Sleep Time	Transmit period	Battery Life
Base Station (Local)	04/05/2010	16:20	Mains	None	5 Seconds	N/A
Fanuc Clean Room(CR) (End Device)	04/05/2010	16:20	AAA (2586)	5 seconds	5 seconds	Approx. 36.5 hours
Battenfeld Clean Room(CR) (Router)	04/05/2010	16:20	AAA (2586)	5 seconds	5 seconds	Approx. 36.5 hours
Router Node	04/05/2010	16:20	AAA (2517)	None	5 seconds	Approx. 2.5 hours

Table 5.1 Battery life test results.

Test two: In this test, the software was left running in the Polymer MNT Laboratory almost continuously for one week only to be stopped momentarily each day to check if everything was fine. The battery life was monitored once again and the results are shown in Table 5.2. After analysing the logged data

Node Name and Type	Date	Time	Power Source	Sleep Time	Transmit period	Battery Life
Base Station (Local)	21/05/2010	16:20	Mains	None	5 Seconds	N/A
Fanuc Clean Room(CR) (End Node)	21/05/2010	16:20	AAA	5 seconds	5 seconds	Approx. 7 days
Battenfeld Clean Room(CR) (End Device)	21/05/2010	16:20	AAA	5 seconds	5 seconds	Approx. 7 days
Router Node	21/05/2010	16:20	AAA	None	5 seconds	Approx.19.5 hours

Table 5.2 Battery life test (over longer time period) results.

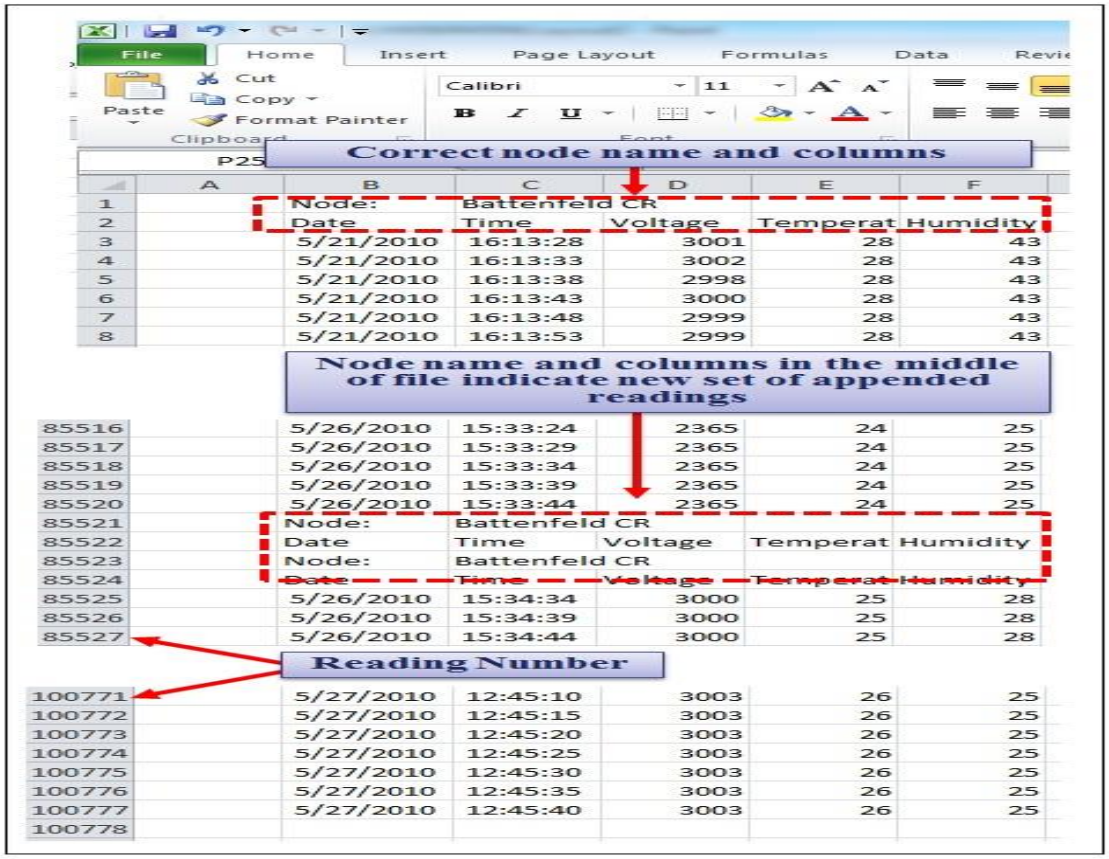


Figure 5.49 Large data log file overview.

clear trends could be identified in the clean room environments. For example when the air conditioning was turned on in the morning time, the humidity in the clean room would start to rise and vice versa, Figure 5.49 shows an overview of the log file for the Battenfeld clean room. The size of the log files was approximately 4MB and the whole weeks' worth of data was analysed for each sensor node. From the resultant files, it could be seen that the logging function was working as expected and that more than a 100K readings were logged in

this period. Every time the Write to File button was clicked to stop and then start the logging each day, the new set of data was appended to the file. This result shows that the LVWSM Systems data logging facility was working as expected and that it could handle large sets of data.

5.6.5.3 *Battenfeld µIM Machine Monitoring using LVWSM System*

In this test, the Battenfeld Microsystem 50 µIM machine was monitored by using the machine interface to connect the machine sensor node as shown in Figure 5.7. The aim of this test was:

- To test the LVWSM Systems ability to monitor the outputs from Battenfeld µIM Machine.
- To test the LVWSM machine nodes GUI while continuously monitoring the machine.
- To test the logging capability of the LVWSM system when using the machine node.

The machine node was programmed to send data readings every second to the Coordinator and was connected to the Battenfeld Microsystem 50 µIM machine using the hardware interface. The machine was left running and the data from the machine was collected using the LVWSM system.

5.6.5.3.1 **Battenfeld µIM Machine Monitoring using LVWSM Results**

The machine was left running for approximately one hour and the data from the machine sensor node was collected using the LVWSM system. Figure 5.50 shows the output for piston velocity and displacement. From this result, the outputs from the machine were being displayed as expected. The piston displacement could be seen changing when the injection piston was injecting the material into the mould. The velocity of the piston was also being displayed and was as expected. Figure 5.51 shows the output for the cavity pressure and nozzle pressure and from this result, the outputs were being displayed as expected. The nozzle pressure could be seen changing when the material was injected into the mould. The mould cavity pressure could also be seen changing as more and more material was being pushed into the cavity. Figure 5.52 shows an overview of the resultant log file created by the LVWSM logging utility. From

this result, it can be seen that the data values are being logged as expected. For this test, approximately 33K readings were logged in the file. The node name and column headings were displayed at the beginning of the file as well as being displayed in the middle of the file indicating new set of data readings have been appended. Although the data logging and data monitoring using GUI capability of the LVWSM system was working as expected, an analysis of the logged data indicated that the machine data resolution was not very high and was around average 15Hz. This resolution will be fine for the general monitoring the parameters but it will not be adequate for process characterisation as this requires higher sampling resolution.

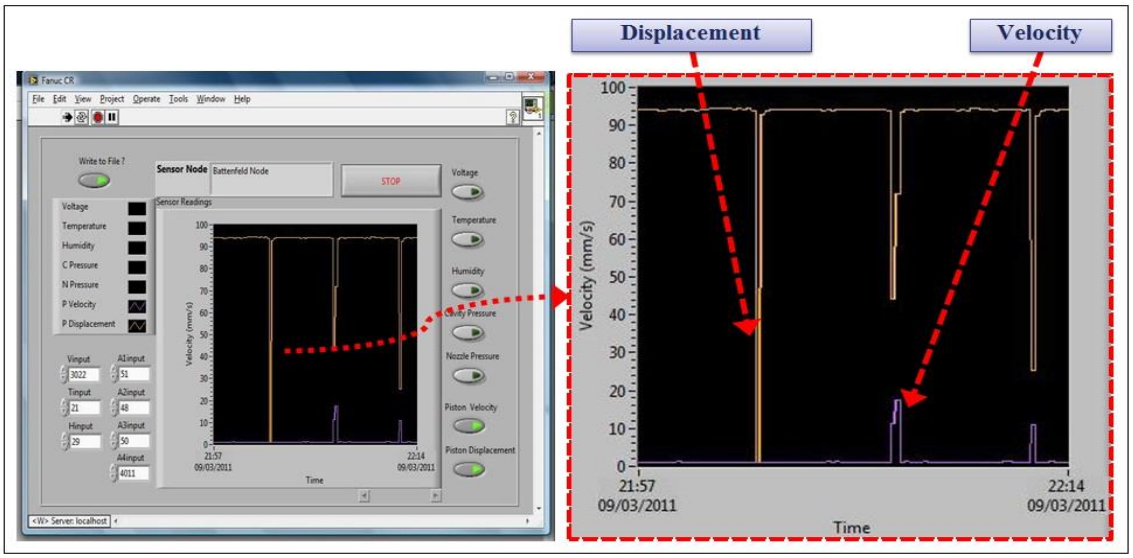


Figure 5.50 µIM Machine injection piston monitoring using LVWSM System.

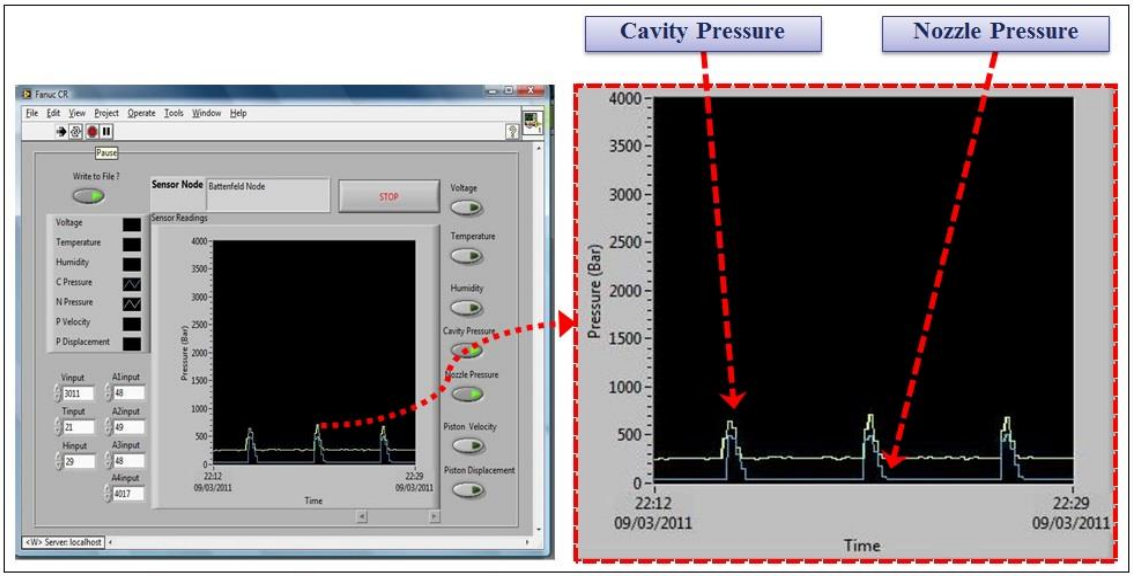


Figure 5.51 µIM Machine pressure monitoring using LVWSM System.

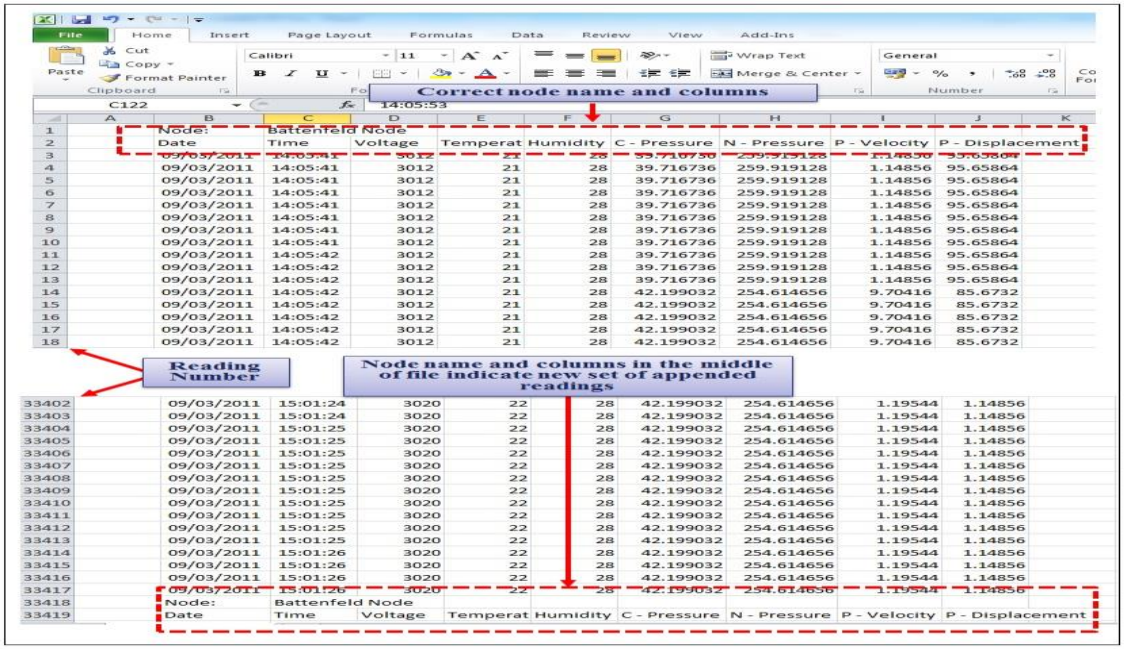


Figure 5.52 Log file for machine sensor node.

5.6.5.4 *Machine Node Data as a Web Service Test*

In this test, the RESTful Web Services created by the Sensor Web layer were deployed on the local LabVIEW Web Server. The Web Service can be accessed using HTTP request to the correct server, followed by the server port and the actual LabVIEW Web Service being invoked. So in the case of the machine node the Web Service deployed is called “machineWS” and the last part of the address of the node is “110955” Therefore a HTTP request in this case to access this Web Service is http://localhost:8085/machineWS/110955. The data being returned was previously shown in Figure 5.46 and is in XML format displayed in a web browser. The aim of this test is to use the deployed Web Services to integrate the machine data with web-based applications. This will be implemented in two scenarios. First using the LabVIEW Web UI builder application and then using the Presto mashup application.

LabVIEW Web UI: In this part of the test the LabVIEW’s Web UI builder application was used to design a simple web interface for the created Web Service. The Web UI interface application used a loop to continuously retrieve data every second. Figure 5.53 shows the Web UI interface application displaying the machine node data by invoking the deployed LabVIEW Web Service.

Jackbe Presto Mashups: The second scenario used Web Mashups to display the machine data. This was a relatively new technique on the web for accessing and merging data from various sources at that time. A simple mashup was created using the Presto Web Mashup application builder. This mashup invoked the deployed LabVIEW machine Web Service and the retrieved data was displayed online in HTML as shown in Figure 5.54. In this test, the ability of integrating the machine data with web-based applications using the LabVIEW machine Web Service was demonstrated. The results showed that the monitoring of the μIM process on the web is quite feasible and this could be the way forward for the next generation process monitoring applications.

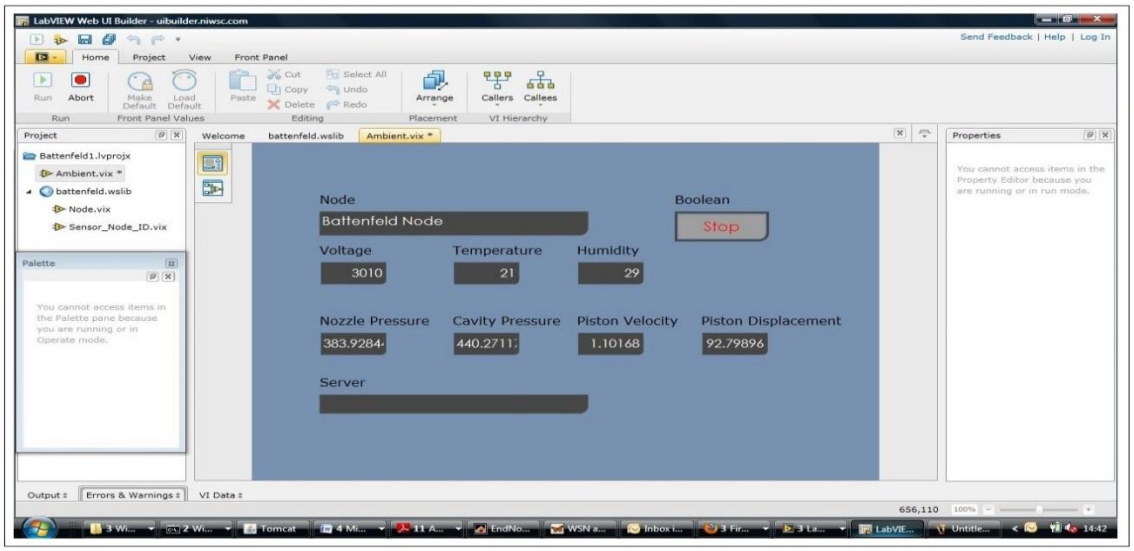


Figure 5.53 LabVIEW Web UI interface application displaying machine data.

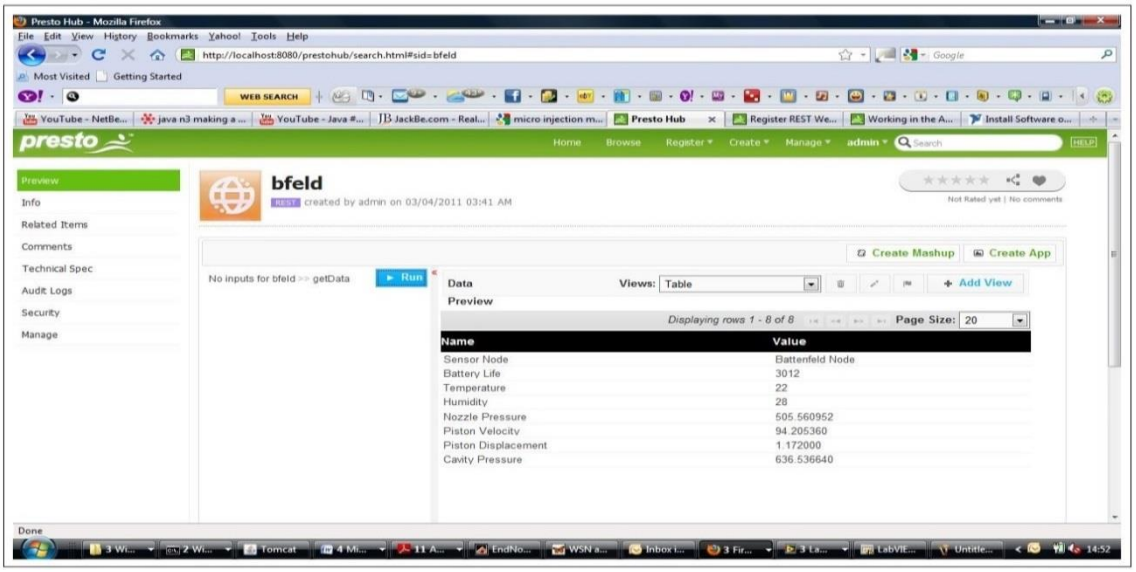


Figure 5.54 Presto Web Mashup application displaying machine data

5.7 Conclusion

While all the previous research focus was on the monitoring of the IM/µIM processes, resulting in a number of customised process monitoring systems that were difficult to integrate or interoperate with other systems and business components of the enterprise. So far, up and until now there has been no focus on making the IM/µIM process or parts of the process available to the outside world or integrating with other business components of the enterprise using web technology. In this chapter the complete architecture, design, development, and implementation of the LVWSM µIM process monitoring system was presented which tried to address the above integration issue through the use of WSN and Web Services. The chapter focused on two distinct parts of the system: WSN hardware programming and interface; and the LVWSM µIM process monitoring application.

To monitor the µIM machine process parameters, the Jennic WSN node firmware was modified to implement a state machine to read the analogue ports and convert the analogue signal into digital form using the on-board ADC. A simple hardware interface was designed to connect the machine sensor node to the machine. The dynamic nature of the WSN such as nodes joining and leaving; transmitting live data from various locations; required the creation of a novel application which could take these requirements into consideration. The solution was the novel LVWSM architecture, which used the concept of dynamic application instance using application templates. This had further challenges of sorting the dynamic data coming from various sensor nodes at the same time. A novel data parser was developed which used data token strings to overcome this challenge. Two types of sensor node templates were developed to cater for the different types of nodes. The use of dynamic instances and templates made this architecture scalable hence making the integration of WSN from other manufacturers easier in the future. The Sensor Web layer was developed to allow the data from the µIM process to be converted into Web Services hence allowing integration with other web applications and components of an enterprise.

The testing of each layer was done separately and the complete prototype system was further tested in a number of scenarios. Results from

these tests showed that the LVWSM µIM process monitoring software had the ability to detect dynamic nature of the nodes as well as displaying the live data from the process with options of viewing individual sensor data. The data could also be logged by the software in a file format as well as in a relational database. The system had the ability to monitor the polymer industrial environment as well as the µIM machines at the same time. During the testing of the complete software a number of minor issues were detected and rectified. One key issue that came to light with this system was the resolution of the machine data was low around an average of 15Hz. This resolution will be fine for process monitoring purposes as well as the environemntal variables but it will not be sufficient for the kind of resolution required when monitoring high pressures and temperatures for the purpose of process characterisation. The other problem which came to light was that the WSN nodes could only send a maximum of about 250Kb per second in theory but in reality this was around 120Kb. Therefore when dealing with high-resolution data this would be a problem and required data storage on the sensor node

Overall, the system was found to be excellent for monitoring the low-resolution environemntal data but for high-resolution data further developments in the hardware such as higher resolution ADC and on-board storage were required. A number of other options are available; one is with latest developments in WSN technology to use an alternative WSN which can cater for the above shortcomings or the other option is to use an altenative data acqusition technology such as Compact-RIO from national instruments for high-resolution data. With the later option this would mean the development of a new architecture which would allow both the current LVWSM system and a high-resolution system to coexist and interoperate. In chapter six the design and implementation of such an architecture using latest web technologies is presented.

Chapter 6: Design of an Enterprise Service Bus (ESB) and Google Gadgets (GGs) Based µIM Process Monitoring System

6.1 Introduction

In the previous chapter the design and implementation of a WSN (ZigBee), based process monitoring system was presented. While other short-range wireless communication protocol standards such as Bluetooth (IEEE 802.15.1), Ultra-Wideband (UWB, IEEE 802.15.3), and Wi-Fi (IEEE 802.11) were available for this purpose, ZigBee was chosen for a number of reasons. Although Wi-Fi and UWB supported higher data rates but from previous research, ZigBee was found to be easy to configure, low cost, self-organising, self-healing, multi-hop, and reliable mesh network. In addition, it had the ability to support over 65000 nodes in the network at any one time. For the factory of the future that could potentially have thousands of sensors and actuators this would be ideal. At the time of this research, the UWB protocol was still in its early stage of simulation and testing and there were no products available on the market. Another reason for choosing ZigBee based devices is that they are low power and hence have longer battery lifetime. Therefore, in the future when using battery powered nodes to monitor inaccessible/hazardous areas on the factory shop floor, the ZigBee devices would last almost twice as long compared to a Wi-Fi node. Finally, from an application point of view, ZigBee was designed for reliable wireless network monitoring and control, while Wi-Fi is directed at computer-to-computer connections as an extension or substitution of cabled networks.

The test results from the LVWSM system showed that the system was excellent for monitoring the low-resolution environmental data. However, for high-resolution machine data, further developments in the hardware were required. To address these shortcomings a number of options were available. The first option was to use an alternative WSN that can cater for the above deficiencies. Even if another WSN was used with a higher resolution ADC, the problem of data transmission would still exist. The ZigBee WSN nodes can only transmit a maximum of 250Kb per second, therefore when dealing with high-

resolution data this would require data storage on the sensor node. The second option was to modify the WSN node hardware or to further develop the hardware interface to include external ADCs and storage devices. The final option was to use a separate high-resolution data acquisition technology and integrating it with the LVWSM system. This option seemed to be the most viable in terms of future extensibility and scalability. To allow the LVWSM system and the high-resolution data acquisition system to operate as one integrated entity would require interoperability between these systems and any other future systems. Therefore, an extensible and scalable integration architecture is required to ensure the interoperability amongst the disparate systems.

This chapter proposes the µIM Process (µIMP) Monitor system based on the middleware layer approach for monitoring a typical plastics machinery environment using WSN, SOA, and Google Gadgets (GGs) (Google Inc., 2011). The middleware layer adopts a service-oriented approach based on the SOA philosophy, where services are independent resources and their implementation details are hidden behind the service interface. The Enterprise Service Bus (ESB) is used as the enabling middleware technology for implementing the SOA. This approach uses messaging and asynchronous communication to remove the integration complexity from the different systems involved and is applied to a heterogeneous network of WSN nodes and National Instruments (NI) high-speed data acquisition devices. The technical issues relating to the system architecture design, component integration, user interfaces and the resulting services are discussed. A prototype system developed for the centre for Polymer MNT at the University of Bradford is discussed which makes use of ESB at the core of the WS02 Carbon Platform's SOA suite of tools (WSO2 Inc, 2011c).

This chapter is organised as follows: Section 6.2 introduces the proposed µIMP Monitor system and its functional architecture for integrating high-resolution (machine data) systems with the LVWSM. It presents the design of the proposed novel architecture based on three major technologies: WSN, ESB, and GG. Section 6.3 looks at the ESB middleware layer in more detail and presents its functional architecture. Section 6.4 gives an overview of the mechanisms of message based communication between the different layers of

the µIMP Monitor project. Section 6.5 presents the design of WSO2 ESB based middleware layer services and artefacts. Section 6.6 presents the implementation and testing of the proposed architecture using GGs API and WSO2's web visualisation and charting library (WSO2 Inc, 2012c). This library has been built upon the Stanford Universities visualization tool called Viskit. Finally, section 6.7 presents the conclusion of this chapter.

6.2 µIM Process (µIMP) Monitor System Design

In the previous chapter, the LVWSM system was found to be excellent for monitoring the plastics machinery environment but not adequate enough to monitor the µIM machines due to the limitations in WSN hardware. Therefore, there is a need for a separate system, which supports high sampling rates in order to adequately monitor high-resolution machine data. Here the National Instruments (NI) reconfigurable embedded controller and acquisition system called Compact-RIO (National Instruments, 2012) was tested for this purpose as it was specifically designed for use with LabVIEW. The Compact-RIO can potentially support sampling rates of 10 KHz and above and therefore this was more than adequate for machine data characterisation. The data from the high-resolution system will also be converted into Restful Web Services. The Web services can be hosted locally on the Compact-RIO embedded Web Server or on a remote machine over the ether network. Further information on the Compact-RIO can be found at (National Instruments, 2012d). In order to allow the LVWSM system, the high-resolution Compact-RIO data acquisition system as well as any other systems to operate as one integrated entity, interoperability between the systems is required. Therefore, a new integration architecture is required to ensure the interoperability amongst the disparate systems. This architecture needs to be versatile to support heterogeneous network systems as well as extensible in order to allow future systems to be integrated. It also needs to be scalable to cater for the increasing number of services without having any effect on the overall performance of the integrated system. This will in effect be the middleware.

In this section the µIM Process (µIMP) Monitor system is presented which is based on a novel architecture using the middleware layer approach. The

Enterprise Service Bus (ESB) is used as the enabling middleware technology for implementing the SOA. The middleware layer approach using the ESB has been used in a number of research studies in order to allow multiple distributed IT systems to interoperate. (Kotsopoulos et al., 2010) in his thesis presented an ESB based architecture to allow distributed local and global network management systems to interoperate. (Tu Quach et al., 2010) used the ESB to allow multiple local and global air pollution management systems to interoperate. The study by (Garcs-Erice et al., 2009) has looked at integrating different WSN platforms under one unified platform using the ESB. This study uses a publish/subscribe middleware to interconnect different WSNs and does not apply this approach to high-resolution DAQ devices and other disparate hardware in the plastics industry environment. To the best of the author's knowledge, no one has so far implemented the middleware layer approach using the ESB to allow embedded hardware based systems to interoperate and monitor the plastics machinery environment. The following sections present the architectural design, system integration and user interfaces for a prototype system developed for the Centre for Polymer Micro and Nano Technology (MNT) at the University of Bradford, which makes use of the WSO2 Carbon Platform's Enterprise Service Bus (ESB) and the GG API.

6.2.1 The µIM Process (µIMP) Monitor Middleware Layer Functional Requirements

To enable the LVWSM system to interact with high-resolution data acquisition system or any other future systems, a middleware layer based architecture needs to be defined in order to provide interoperability between the different systems. Before deriving the functional architecture of the ESB based middleware layer, its functional requirements need to be defined. In order to determine what the functional requirements of the µIMP Monitor system would be, the systems being integrated as well as the services being provided by these systems need to be analysed.

LVWSM System: The LVWSM system uses a WSN to collect data from the environment and converts the sensor readings into Restful Web Services. The WSN uses the ZigBee protocol. The hardware nodes placed in the plastics

machinery environment use embedded microcontrollers, which run the ZigBee stack firmware.

LabVIEW Simulation System: The LabVIEW simulation system is made up of a two simple applications written in LabVIEW. These applications were primarily created for testing the architecture without the hardware being connected. The first application simulates a random data generator system, which outputs random data values between two threshold points at specified time intervals. The random data is then converted into a Restful Web Service to be consumed by web applications. The second application simulates a counter system, which counts up from zero to a user specified value and then counts down to zero. The counter data is also converted into a Restful Web service. The LabVIEW simulation system resides on a standard Intel i7 PC system and is connected to the local ether network. The LabVIEW Simulation System was developed to test the µIMP Monitor system before it is connected to live WSN and machine data.

LabVIEW Compact-RIO Application (LVCRA) System: The high-resolution LVCRA system will be based on the NI Compact-RIO system, which has an integrated embedded controller as well as an FPGA backplane. The FPGA is used to trigger the data acquisition in almost real-time and the embedded controller is used to acquire the machine data at high sampling rates. The data is processed and made available to a LabVIEW application residing on remote machine, which converts the data into Restful Web Services. The LVCRA runs on an Intel i7 PC machine connected to the local area network.

Web Application System: The data from the different systems needs to be made available on the internet using Web Services and standardized message based communication. Hence allowing all services to use the same format and transport protocol. The Simple Object Access Protocol (SOAP) is the standard transport protocol for messages processed by Web Services (W3C, 2007a). This protocol exchanges XML-based messages over a computer network, using Hypertext Transport Protocol (HTTP) or Java Messaging Service (JMS). The ESB will expose service endpoints that accept SOAP messages from clients.

These services act as proxies for external services and the ESB mediates the messages before they are proxied to the actual service. Web applications such as GG need to be developed to consume and merge the various Web Services created by the systems under a single unified web portal.

From the above analysis the middleware requirements can be generalized into six categories (Pinus, 2004, Kotsopoulos et al., 2010), as shown below:

- **Heterogeneity:** The middleware should have the ability to support heterogeneous hardware and software platforms such as the ones described earlier.
- **Network communication:** The middleware should allow the communication between heterogeneous networks used in the above systems, regardless of their underlying transport protocols.
- **Mediation:** The middleware should allow the mediation and coordination of information exchange between the heterogeneous systems and services.
- **Reliability:** The middleware must be reliable and should ensure that the data is guaranteed to reach its destination complete, uncorrupted and in the order it was sent.
- **Scalability:** The middleware should be able to accommodate future network and service expansion.
- **Extensibility:** The middleware must be able to accommodate future heterogeneous applications and systems.

The above middleware requirements, combined with the earlier analysis of the systems being integrated, should govern the type of services to be supported in the middleware to allow communications between these systems and services.

6.2.2 µIMP Monitor System Architecture Design

The proposed system architecture of the µIMP Monitor is shown in Figure 6.1. The architecture consists of four layers; µIMP Hardware Layer, µIMP Local Monitor Layer, µIMP Global Monitor Layer, and µIMP Web Layer. This architecture is based on the SOA and middleware layer concepts using the ESB and takes into consideration the heterogeneous nature of the hardware and software platforms used. The raw environmental and machine data is recorded

using the heterogeneous hardware in the µIMP Hardware layer. This is then processed and converted into Web Services by the local applications residing in the µIMP Local Monitor layer. The µIMP Global Monitor Layer through the use of the ESB middleware transforms and mediates the services to be passed onto web applications residing in the µIMP Web Layer. The following sections describe the four layers in more detail.

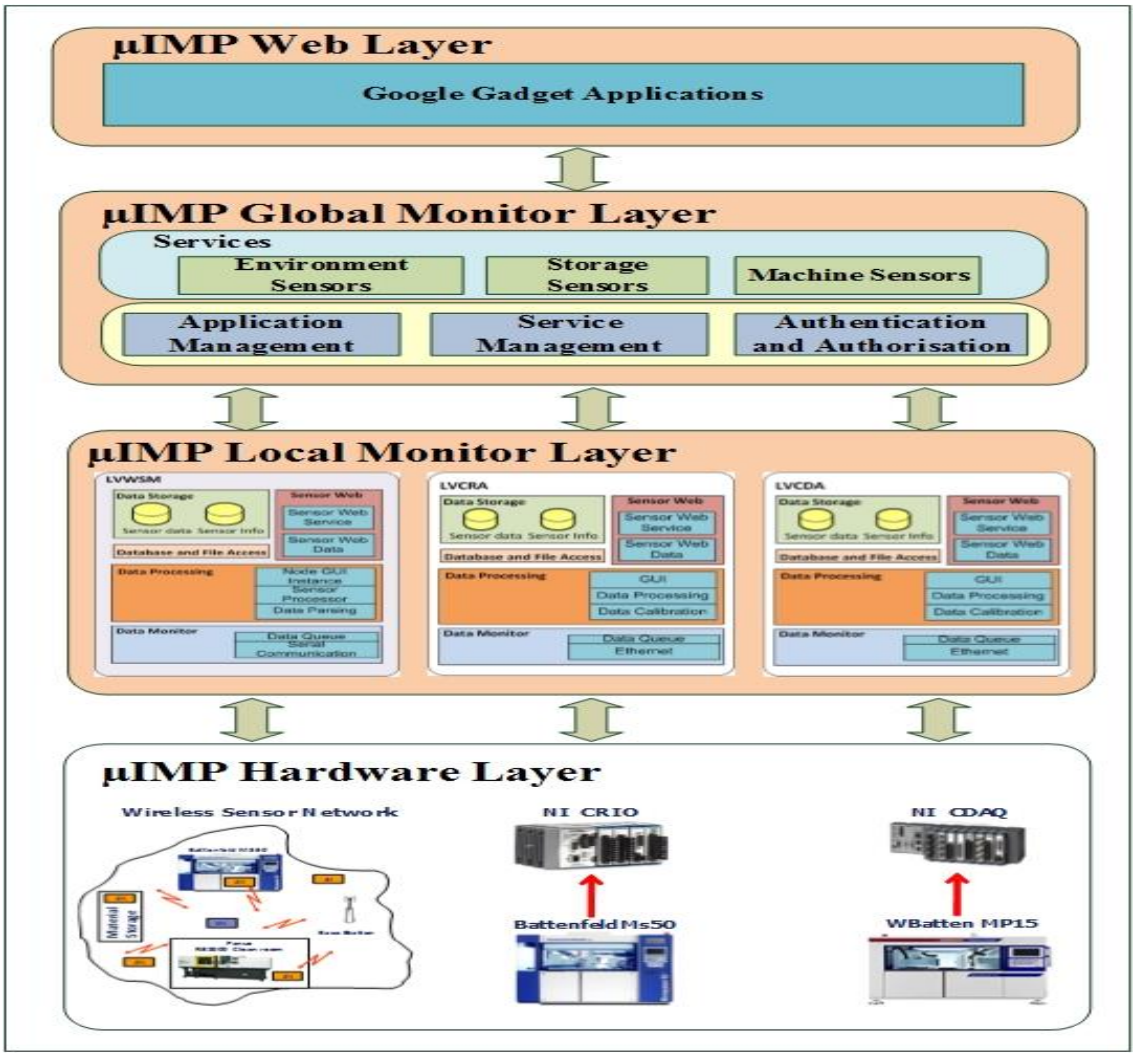


Figure 6.1 µIMP Monitor System Architecture.

6.2.2.1 *µIMP Hardware Layer Design*

The µIMP Hardware Layer uses a heterogeneous network of wireless sensor nodes and Ethernet based NI High speed DAQ devices. The wireless sensor nodes are used by the LVWSM system to record and transmit environmental data such as material storage conditions, ambient lab temperature, humidity and light. The WSN use the ZigBee protocol to communicate with each other and the collected data is passed to a Coordinator node connected to a central

gateway PC running the LVWSM application. To monitor a polymer industrial environment, the wireless sensor nodes need to have the following minimum specifications. An 8-bit microcontroller running at 16MHz, with 4Kbytes of SRAM memory, 2.4GHz radio transceiver, on-board ambient sensors, on-board ADC and DAC (10bit), and on-board digital/analogue ports for external sensor interfacing To monitor the IM/µIM machines requires high specification monitoring due to the high pressures (>2000 Bar) and injection speeds (<10ms cavity filling times). With typical sampling rates being above 1 KHz means that there will be a large amount of data to be processed. The high-end sensor nodes from NI with a processing power of 533MHz and external flash RAM will be able to meet the processing requirements to a certain extent but they still would not be sufficient to characterise the µIM process. Therefore, the sensor nodes required for this purpose need to be modified by adding higher resolution ADCs, on-board memory, external flash memory, and high bandwidth transceivers or use alternative technologies. The high-resolution process data

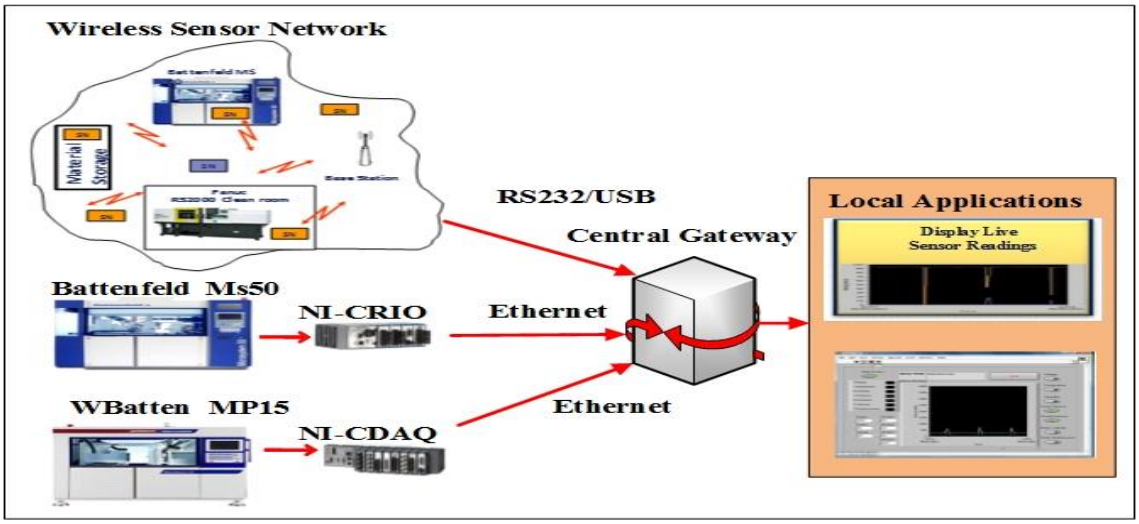


Figure 6.2 Heterogeneous network of sensor devices communicating with a Central Gateway.

from the µIM machines is recorded using an alternative technology the NI DAQ devices (Compact-RIO and Compact-DAQ) at acquisition rates in excess of 1000Hz and transmitted over the local ether network. A central gateway (Figure 6.2) processes, stores, converts and presents the data as a service using Web Services. Using a remote server, this data is then integrated with other process information related to the specific product.

6.2.2.2 *µIMP Local Monitor Layer Design*

The applications that record the data from the different hardware reside in the µIMP Local Monitor Layer. This layer is responsible for recording, formatting and converting the raw data from the various sensors in the environment and machines using the local applications. The raw data collected from the various sensor nodes in the WSN is recorded using the LVWSM application. The high-resolution machine data is recorded using NI High speed DAQ devices in conjunction with the LVCRA and LVCDA applications, developed using NIs LABVIEW graphical development environment. In this layer, the data from all the applications is recorded in a CSV file format as well as a MySQL database and serves as a benchmark for monitoring and detecting trend patterns. The data is also converted into Web Services to be consumed by web applications and processed in real-time for live update of the process. The sensor data applications have a common base architecture made up of four components as shown in Figure 6.3; Data Monitor, Data Processing, Data Storage, and Sensor

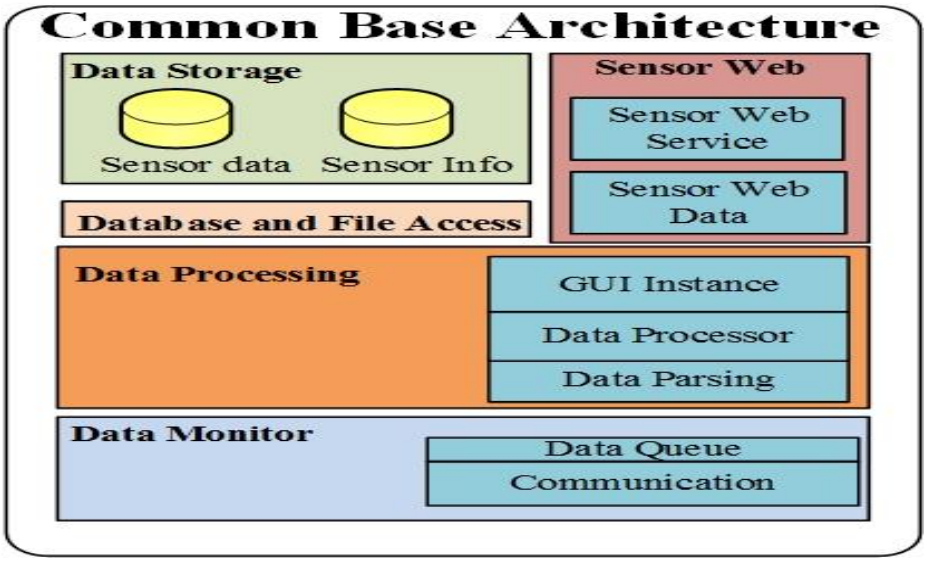


Figure 6.3 Common Base Architecture for Local Layer Applications

Web. The Data Monitor component is responsible for monitoring the communication ports for incoming data between the application and the hardware. The Data Processing component formats live data from the communication buffers and presents it for storage and further conversion. The Data Storage component allows the data to be logged in a file as well as a MySQL database. The Sensor Web component converts the data into a RESTful Web Service hence making it available to be consumed by web

applications. The LVWSM application has extra data parsing and node detection components for each node in the network.

6.2.2.3 *µIMP Global Monitor Layer Design*

The applications in the µIMP Local Monitor Layer can potentially give rise to a large number of services and composite services driving a large number of other services, business processes and even applications. The resulting plethora of information extracted using services from the various sensors in the environment and machines can become difficult to manage and hence requires a middleware enabled composition technology such as an ESB to manage it globally. Using the ESB at its core, the µIMP Global Monitor Layer is the common data communication and messaging layer between the different applications and has three main functions: application management, service management, and user authentication and authorisation. The subsequent sections on the functional architecture will give a detailed description of the ESB based µIMP Global Monitor Layer.

6.2.2.4 *µIMP Web Layer Design*

To monitor the µIM process on the web, applications, which consume web services, are required. The GGs API (Google Inc., 2011) was selected for this purpose. GGs are mini-applications built using HTML/XML with JavaScript, Flash or Silverlight for dynamic behaviours. Although the GG is used in this project, various other web-based applications that consume Web Services can be used for this purpose. One of the main advantages of using GGs is that mini applications can be developed for the different stages of the µIM process hence allowing the development of more complex applications in the future which, are modular in nature.

6.3 WSO2 ESB based Middleware Layer Functional Architecture Design

To accurately integrate the complete µIM process lifecycle (Figure 6.4) from the point of material purchase to the finished end product delivered to the customer requires a unified method of integration. This integration method will have to merge the various systems running in the enterprise and allow them to interoperate using a common communication medium. In order to allow the

complete process to be monitored on the internet the disparate hardware and software components of the µIM process lifecycle need to be integrated with web-based technologies such as the GGs Gadgets API. The need arises for a unified method of data integration between these components via a unified web-based portal. The data from the different components can be in string, integer and binary format and needs to be manipulated effectively. To meet this integration requirement, the ESB was used, an open framework, which provides an implementation backbone for an SOA that treats applications as services. It helps to compose, deploy and coordinate communications between service components of a distributed system via standards-based adapters and interfaces. With endpoints, the ESB provides flexibility in the transport layer by enabling loose coupling and easy connection between these services. As well as facilitating application and process integration through messaging control, intelligent routing, and dynamic data transformation, the ESB also applies the security, policy, reliability and accounting needs required in an SOA. For the µIMP Monitor, the lightweight and open source WSO2 ESB is selected which is based on Apache Synapse mediation engine.

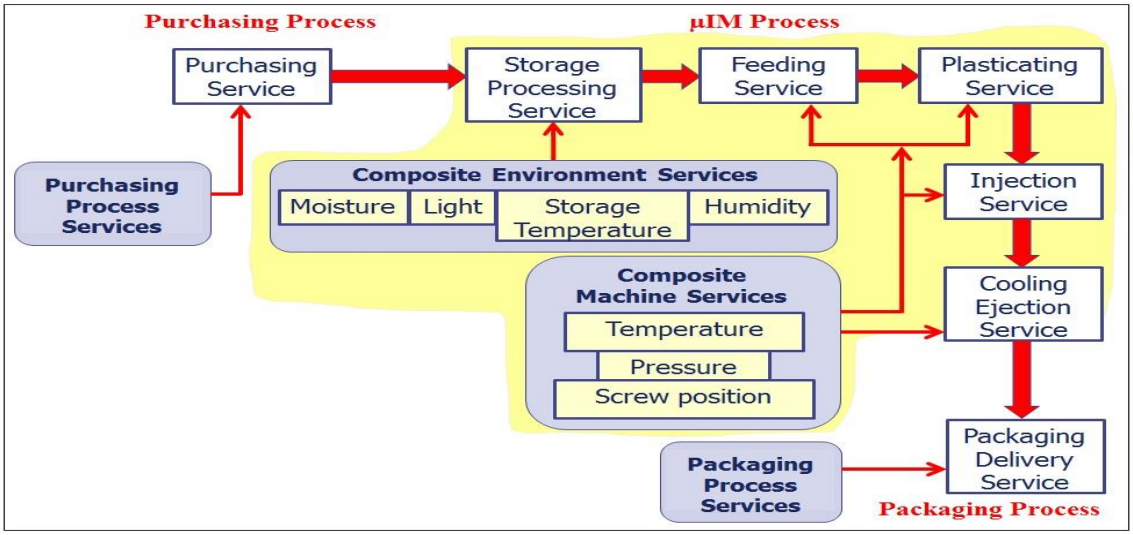


Figure 6.4 A Complete µIM Process Lifecycle.

6.3.1 WSO2 ESB Architecture Design

The WSO2 ESB is a fast, lightweight and user-friendly open source ESB developed on top of the WSO2 Carbon enterprise middleware platform. This middleware is based on the Open Services Gateway initiative (OSGi) (OSGi Alliance, 2011) to provide better modularity and componentization to the SOA.

The WSO2 Carbon platform includes more than 175 components that deliver; messaging, data, business, presentation, identity, security, governance, monitoring, and management services (WSO2 Inc, 2011c). The WSO2 ESB allows users to easily configure; message routing, intermediation, transformation, logging, task scheduling, failover routing, and load balancing. It also supports; transport switching, event, rule based mediation and priority based mediation for advanced integration requirements. The ESB runtime is designed to be completely asynchronous, non-blocking and streaming.

6.3.2 µIMP Monitor Functional Architecture Design

Figure 6.5 shows the proposed functional architecture design of the µIMP monitor with the WSO2 ESB at its core. The WSO2 ESB has been proposed as the unique method of communication between the different applications in the µIMP Monitor project. The ESB will reside in the µIMP Global Monitor Layer to allow backend hardware and applications to communicate with frontend web-based applications such as GGs.

The process monitoring applications in the µIMP Local Monitor Layer use three distinct hardware platforms (WSN, Compact-RIO, Compact-DAQ) with unique embedded architecture and firmware. The µIMP Web Layer on the other hand has mini applications written using the Google Gadget API. The WSO2

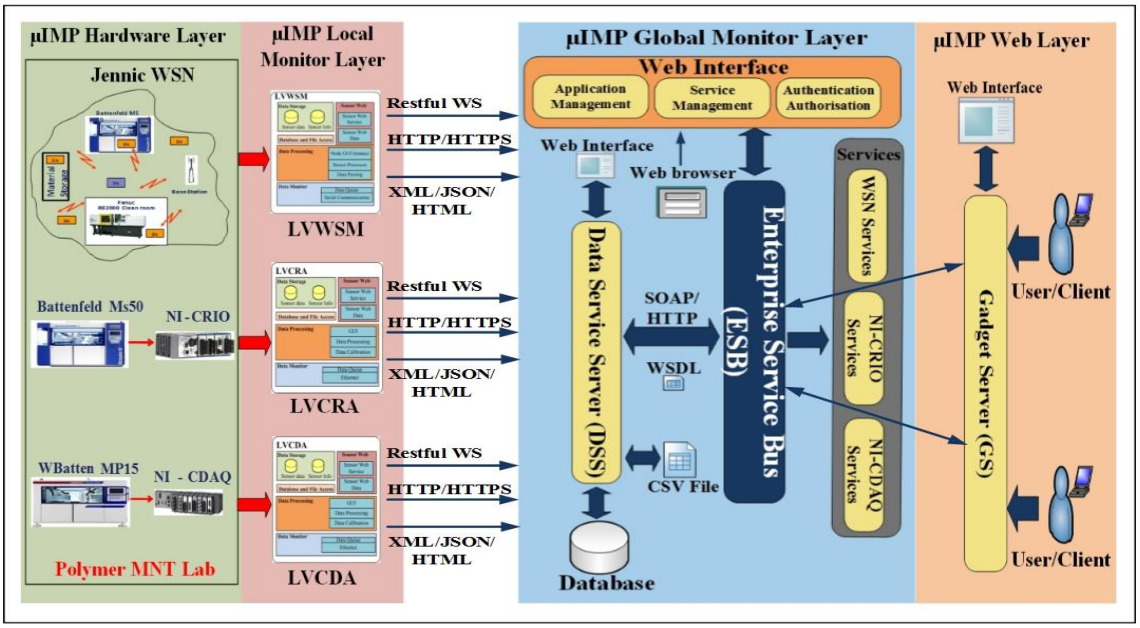


Figure 6.5 ESB based µIMP Monitor Functional Architecture.

ESB acts as the shared messaging layer between the different backend applications and services of the µIMP Local Monitor Layer and the frontend GG web applications residing in the µIMP Web Layer. It uses intelligent transformation and routing to ensure reliable message delivery to the correct service or application. The µIMP Monitor system will provide a number of services for each hardware device. The WSN has three services for each node: temperature, humidity and voltage. The NI Compact RIO (NI-CRIO) and NI Compact DAQ (NI-CDAQ) for each machine provide nozzle pressure, cavity pressure, temperature, piston displacement and piston velocity. These services are created using RESTful web services (Pautasso et al., 2008). The RESTful web services apply Roy Fieldings REST design principles (Fielding, 2000) to develop web services. RESTful Web services focus on the Web resources as the main abstraction.

6.3.2.1 *µIMP Hardware Layer Functional Architecture Design*

The µIMP Hardware Layer consists of various hardware platforms, which are connected to the relevant applications in the µIMP Local Monitor Layer. Depending upon the hardware used and how the SOA paradigm has been deployed on the actual hardware (deployed at the device level or on the gateway middleware) this layer makes available the device level services or raw data to be used by the µIMP Local Monitor layer applications. In this thesis, the SOA is deployed at the gateway middleware hence resulting in raw sensor data.

6.3.2.2 *µIMP Local Monitor Functional Architecture Design*

The µIMP Local Monitor Layer exposes the raw data (recorded using the various hardware platforms) using local applications. The local applications used in µIMP Monitor project use a common base architecture described in section 6.2.2.2. These applications format the raw sensor data into file and database formats as well as converting it in Restful Web Services. Each application in this layer creates services that support multiple message formats (JSON, XML, HTML, and Text) and transport protocols (HTTP, JMS). In this project, XML messages are used over HTTP/HTTPS transport.

6.3.2.3 *µIMP Global Monitor Functional Architecture Design*

The µIMP Global Monitor Layer consists of four main components; the Data Service Server (DSS), the ESB, the Web Interface, and the Service registry.

Data Service Server (DSS): The DSS is based on the WSO2 Carbon platform (WSO2 Inc, 2011b) and runs on a separate PC machine. The data collected from the process monitoring applications in the µIMP Local Monitor Layer is stored in CSV file format as well as MySQL database. The DSS server has the ability to connect to external files and databases and its main purpose is to create data services from the data stored in these files and databases. The data services created by the DSS are used by the GG web applications to serve as a benchmark for monitoring and detecting trend patterns.

Enterprise Service Bus (ESB) Server: The data from the process monitoring applications in the µIMP Local Monitor Layer is converted into Restful Web Services to be consumed by ESB server. The ESB server allows the communication between service components of a distributed system via standards-based adapters and interfaces. It consumes the data services from the DSS server as well as the Restful Web Services from the process monitoring applications. Proxy services are created for each external service and the ESB exposes service endpoints that accept messages from clients. These services act as proxies for external services and the ESB mediates the messages before they are proxied to the actual service. When a message is received from an external service, it goes through the ESB mediation engine which performs any mediation specified in the proxy service such as data transformation and augmentation. Once this is complete, the message then gets sent to the relevant address specified through the use of an endpoint. The WSO2 ESB has a graphical Web Interface as well as the Service Registry, which are described in the following sections.

Web Interface: The ESB Web interface component is responsible for the application and service management as well as user authentication and authorisation. It allows the setup and configuration of customised user web applications as well as the creation of proxy web services. Users can access the ESB using a standard web browser through its graphical interface.

Services Registry: The ESB service registry component runs in the background to allow the services to be registered and discovered in a central services registry. Using the graphical interface the user can create new proxy service using various configurations (mediators, transformations, endpoints) and link them to external web services.

6.3.2.4 *µIMP Web Layer Functional Architecture Design*

The µIMP Web Layer consists of a single component the WSO2 Gadgets Server (GS) running on a separate PC machine. This component is also built upon the WSO2 Carbon platform and uses the GGs API to create and deploy GGs. The GGs use JavaScript and XML/HTML to communicate with Google Gadget API. One of the advantages of using GGs is that mini applications can be developed for the different stages of the µIM process. Linking these applications would allow the development of complex applications, which are modular in nature. The GGs server consumes and sends requests to the proxy services of the ESB server, which in turn communicate with the actual µIMP Local Monitor Layer Restful Web Services, and processes them for live update of the process. The WSO2 GS also has a web interface, which allows the users to create, configure, and setup gadget applications imported from external sources. The WSO2 GS is the frontend web application of µIMP Monitor system and will allow the monitoring the µIM process on the internet using GGs.

6.4 Middleware Layer Message Based Communications

Before going onto describe the different service components and artefacts created for middleware layer this section gives an overview of the mechanisms of communication between the different layers of the µIMP Monitor project.

6.4.1 Message Orientated Middleware (MOM) Concept

The WSO2 ESB middleware layer uses the MOM concept in which applications employ a message client API to communicate with each other through a messaging system. Either the MOM communication paradigm allows applications to act as the message producer that produces (sends) the message or a message consumer that consumes (receives) the message. Applications may have dual functionalities of being a producer and a consumer at the same time. In relation to the µIMP Monitor project, the µIMP Local

Monitor Layer will primarily be the message producers whereas the µIMP Web Layer will be the message consumer, which retrieves the sensor data from the various proxy services linked to the µIMP Local Monitor Layer applications. Although the WSO2 ESB allows users to implement a number of different message brokers (FIX,JMS, Mail etc.), it uses the built in Apache Qpid message broker (Apache, 2012a). The Apache Qpid broker is used to allow communication between the message producers and consumers. The Apache Qpid is a fully developed MOM, supports the two most common models for information exchange through message virtual channels, namely, the point-to-point (Queues), and publish/subscribe (pub/sub) models.

6.4.1.1 *Point-to-Point*

In the point-to-point messaging model, messages from the message producer are routed via virtual channels to the consumer via a queue. This model allows message clients to send and receive messages asynchronously. There is no restriction on the number of message producers who can publish to a queue, a message in the queue can only be received by a single message consumer.

6.4.1.2 *Publish/Subscribe*

In the publish/subscribe model, messages are published according to a certain topic of interest. Virtual channels are used for each topic of interest to which a consumer subscribes its interest. The pub/sub model supports one-to many and many-to-many distribution mechanism, allowing a single producer to broadcast a message to hundreds of thousands of consumers.

6.4.2 **Message Relaying**

Once the message from the producer has been received by the ESB, it needs to be relayed to the relevant consumers. The out of the box WSO2 ESB configuration provides several message transfer mechanisms to address different integration requirements. The WSO2 ESB has the ability to provide efficient message transfer under message streaming mode, it offers two streaming mechanisms, namely, Message Relay and the Pass-Through (PT) HTTP Transport (Jayasumana, 2012).

6.4.2.1 *Message Relay*

In the Message Relay (MR) mode, the ESB uses message Builder and Formatter components to relay the messages coming into the ESB. The message Builder and Formatter components are based on the Apache Axis2, which is the base for Apache Synapse's ESB engine (Apache, 2012d) on which the WSO2 ESB is built upon. It helps users to add their custom message formats through Builders and Formatters (Apache, 2012b).

A message Builder component is used when Apache Axis2 receives a message; it looks up the message builder registered for the content type of the message, and asks it to build an XML message from the request message provided as bytes. A message Formatter on the other hand is the opposite of the Builder and is used when sending a message out. Apache Axis2 looks up the Formatter registered for the target content type, and asks the Formatter to convert the XML message to bytes (Perera, 2009). Figure 6.6 shows a generic scenario implementation using a message Builder and Formatter. In this scenario, the service provider sends a message using a specific content type. The message Builder component uses specific message Builders depending

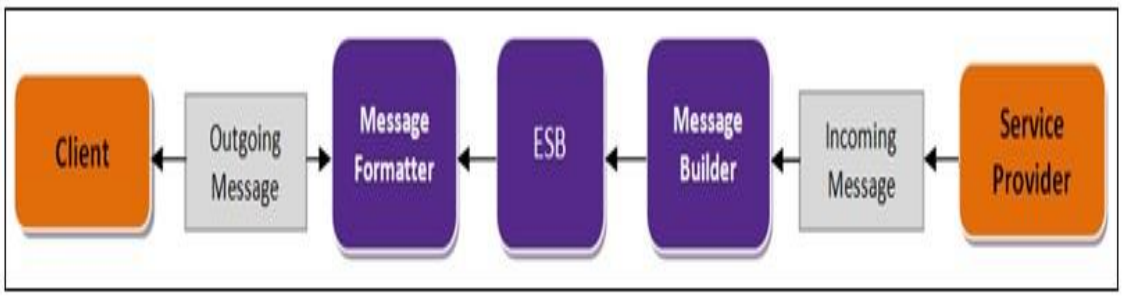


Figure 6.6 A Typical ESB Routing Scenario using Message Builders and Formatters (WSO2 Inc, 2011a).

upon the message content type. The WSO2 ESB uses a few default type Builders if no specific Builder is specified. Once the message has been processed, it is sent to the message Formatter that builds an outgoing stream by looking up a registered Formatter for the target content type and converts the XML message into bytes before sending it to the client.

6.4.2.2 *Pass-Through (PT) HTTP Transport*

In a scenario where there is no message parsing or processing required the WSO2 ESB supports the PT HTTP transport model. This model allows the non-

blocking HTTP (NHTTP) transport to selectively build (or correspondingly not build) the message content. This model is much better than the MR model in terms of message throughput performance as there is no parsing and processing done and the message is passed straight through the ESB. In a pure pass-through scenario where the HTTP protocol is used on both sides and no message parsing or processing is required, it is possible to further improve performance by eliminating one message buffer.

6.5 Design of WSO2 based Services and Artefacts

In order to allow the different layers of the proposed architecture to be integrated a number of services and artefacts were created. The services, which have been created, were mainly for the ESB based µIMP Global Monitor middleware layer, which sits between the µIMP Web Layer and µIMP Local Monitor Layer applications to allow them to communicate with each other. Some artefacts such as the service WSDL files, service endpoint, and mediators were also developed for this layer. The other main artefacts that have been developed include the Google Gadget applications for the µIMP Web Layer. The following sections present the design of these services and artefacts in more detail.

6.5.1 Design of WSO2 ESB based Middleware Layer Services

To allow the WSO2 ESB based µIMP Global Monitor Layer to mediate between the µIMP Local Monitor Layer and the µIMP Web Layer a number of services and artefacts were created. These services and artefacts have been designed and developed using the Eclipse based WSO2 Developer Studio. WSO2 Developer Studio is a complete development platform, which allows the development, deploying, testing and debugging of SOA applications. The WSO2 Developer Studio is the integration of the WSO2 Carbon platform and the open-source Eclipse IDE (WSO2 Inc, 2012d). These services and artefacts are explained in greater detail in the following sections.

6.5.1.1 *Design of DSS Data Services*

The data from each of the µIMP Local Monitor Layer applications needs to be logged into a file and a database. The WSO2 Data Service Server (DSS) will be

used as part of the µIMP Global Monitor Layer to create data services to retrieve and manage the historical sensor data stored in the files and databases. The DSS will need to create two types of data services one for the Excel files and the other for the data to be retrieved from the database. Each data service that is created needs to have a Web Services Description Language (WSDL) file associated with it.

6.5.1.2 *Design of Data Services for Excel Files*

The applications in the µIMP Local Monitor Layer log the various sensor data into CSV/Excel files for the purpose of analysis and trend detections in the process. This sensor data needs to be made available to the frontend Web applications in the µIMP Web Layer via the µIMP Global Monitor Layer. The µIMP Global Monitor Layer will use the DSS to access this data using data services. The data services required will need to be able to read and query the data in the excel file. In the excel file data for each sensor is time stamped and stored in separate columns. Therefore a data service needs to be designed which supports the querying of data in a file using multiple web methods. Web methods need to be created which allow the querying of separate sensor data as well as retrieving all the data and presenting it in an XML format.

6.5.1.3 *Design of Data Services for Relational Databases*

The applications in the µIMP Local Monitor Layer also log the various sensor data into relational databases such as Access and MySQL for the purpose of analysis and trend detections in the process. This sensor data also needs to be made available to the frontend Web applications in the µIMP Web Layer via the µIMP Global Monitor Layer. The DSS in the µIMP Global Monitor Layer will be used to access this data using data services. The data services required will need to be able to read and query the data stored in the relational databases. The data in the MySQL database is organised by using tables for each sensor node. The first column in the table has the date and time stamp and the other columns are created for each sensor on the sensor node. The data services required to retrieve the data from the various tables in the MySQL database needs to support SQL queries and scripts. The service also needs to be implemented using multiple web methods. The required web methods will use

embedded SQL queries to retrieve separate sensor data as well as retrieving all the data and presenting it in an XML format.

6.5.2 ESB Proxy Services

The ESB based µIMP Global Monitor Layer acts as the middleware layer between the frontend Web Applications and backend local applications. This layer will take requests from the web clients and route them to the relevant Web Services of the local applications or data services on the DSS server. The ESB will need to create two types of proxy services depending upon the type of request received from the Web Applications; the first type of proxy service will be for each actual Web Service of the Local applications and the second type for the data services of the DSS. For each proxy service, an endpoint will also need to be created to allow the routing of the messages to the relevant address stored in the endpoint. There will also be the need for creating mediators (a mediator is the basic message processing unit in the ESB) for some of the proxy services if the message needs to be modified or changed. Each proxy service that is created will need to have a WSDL file associated with it.

6.5.2.1 *Design of Proxy Services*

The ESB based µIMP Global Monitor Layer acts as the middleware, mediates, and routes the messages from various applications. It will implement this using proxy services. In an ESB a proxy service is a virtual service that acts as an intermediary between the caller of the service and the actual service. It is used to receive incoming messages and optionally processes them before sending them off to the actual service using a given endpoint. Proxy services allow the users to carry out necessary transformations and add more functionality without changing the existing service. For example, if a service is required to handle messages with different formats, a transformer proxy service can be created to transform requests and responses based on specified eXtensible Stylesheet Language Transformation (XSLT) configuration files. On the other hand, if there is no need to format or transform the service then a pass through proxy service can be created. Other features of Proxy services include the switching of transport protocols, processing messages with mediation sequences and tasks,

and terminating the flow or returning a message back to the client even without sending it to the actual service (WSO2 Inc, 2012b). The WSO2 ESB allows the user to create different types of proxy services and these include:

- **Pass Through Proxy** – This type of service forwards the messages to the endpoint without performing any processing on them.
- **Secure Proxy** – This type of service is mainly used to add security (using WS-Security) to incoming requests before forwarding them to unsecured backend services.
- **WSDL Based Proxy** – This type of proxy service is created by using a remotely hosted WSDL file of an existing web service. The endpoint information is extracted from the WSDL.
- **Logging Proxy** – This type of service logs all the incoming requests from the client and forwards them to a given endpoint. It also can log responses from the backend service before routing them to the client.
- **Transformer Proxy** – This type of service transforms all the incoming requests from the client using XSLT and then forwards them to a given endpoint. It can also transform the responses from the backend services before sending them back to the client.
- **Custom Proxy** – This type of service allows the user to customize the sequences, endpoints, transports, and other QoS settings.

The proxy services that need to be created for the Web Services from the Local Applications will most of the time need to log and forward the message to the relevant Web Services without modifying it to. The actual Web Services will need to retrieve the relevant data and send it back to the proxy service, which in turn sends it back to the requesting GG application. These proxy services will be mainly used for real-time update of the GG application and can be implemented using pass through, logging, or custom proxies.

The proxy services that need to be created for the data services from the DSS server will need to be able to access the web methods defined in the data

services to retrieve sensor data from the files or databases. Therefore, these proxy services will need to access the WSDL file of the data service and can be implemented using customised proxy service or a WSDL proxy service. The data service will need to retrieve the relevant data from the files or databases and send it back to the proxy service, which in turn sends it back to the requesting GG application. These proxy services will be mainly used by gadgets, which have been designed for historical data display.

6.5.2.2 *Design of ESB Endpoints*

In order for the proxy services in the µIMP Global Monitor Layer to route the messages to the right destination, they require a delivery address to which the message needs to be delivered to. This can be achieved by the use of endpoints in the ESB. In an ESB, an endpoint defines an external destination address for a message passing through the ESB. The endpoint may be specified as an address endpoint, WSDL based endpoint, a load balancing endpoint, a fail-over endpoint or a HTTP endpoint. For successful communication between the different layers of the architecture, a number of endpoints will need to be created to be used by the proxy services.

6.5.2.3 *Design of ESB Mediation Sequences*

A mediator is the basic message processing unit in the ESB. Mediators are used to modify and change messages by carrying out some predefined actions before outputting the modified message. In the WSO2 ESB a range of mediators are available for carrying out various tasks on input messages. These include functionalities such as matching incompatible protocols, data formats and interaction patterns across different resources. Functionalities such as data splitting, cloning, aggregating, and enriching are also available, hence allowing the ESB to match the different capabilities of services. An example of one of the most common mediator used is the Send mediator which sends the message to a specified address other mediators include the Aggregate mediator collects and merges the responses before sending them back to the client.

The proxy services that will be created in the µIMP Global Monitor Layer will use various mediators in a mediation sequence. A mediation sequence commonly known as a "sequence” is a number of mediators held in a list and

get executed in order. Sequences are an integral part of the ESB's core and when a message is delivered to a sequence, it sends the message through all the mediators held in the list. Figure 6.7 shows what happens to a message in the mediation process when it arrives at the ESB. When the message arrives at the ESB if it is not destined for a proxy service it is sent through the mediation configuration, which holds two special sequences named “main” and “fault”. By default, the main sequence simply sends a message without mediation but if further mediation is required then new mediators and/or named sequences can be added into the main sequence. If an error occurs in the main mediation sequence (Figure 6.8) the message is passed to the fault sequence, which has its own set of mediators which are used to relay an error back to the client. The most common mediators that will have to be used will be the Log and Send mediators. The property mediator will need to be used in the design of the proxy services used by the actual Web Services of the Local applications. The property mediator will be used to define the type of transport method to be used when accessing Restful Web Services from the µIMP Local Layer applications developed in LabVIEW.

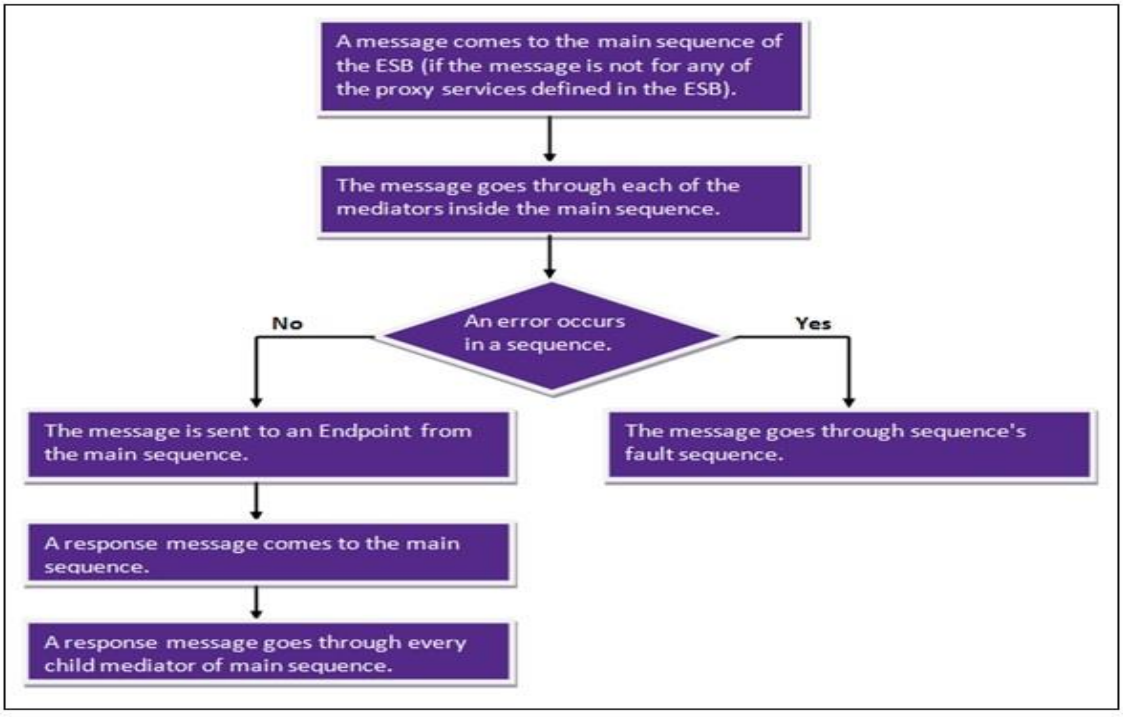


Figure 6.7 The Process of ESB Message Mediation(WSO2 Inc, 2011a).

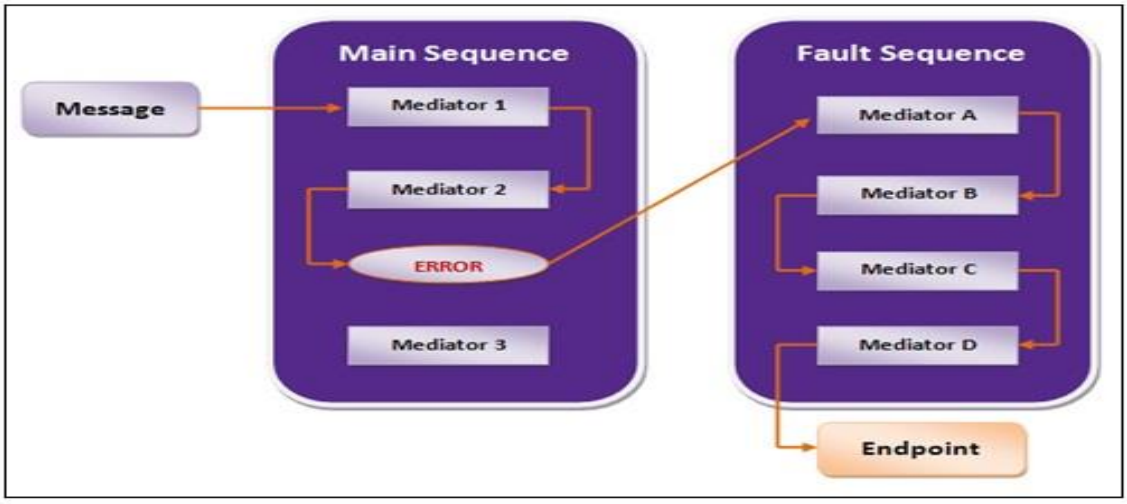


Figure 6.8 Message Mediation Fault Handling (WSO2 Inc, 2011a).

6.5.3 Design of µIMP Web Layer Google Gadgets

In order to allow web-based applications and clients to communicate with the µIMP Local Monitor Layer applications linked to the backend µIMP Hardware Layer; the µIMP Web Layer uses web-based GG mini applications. The GGs are written using XML/HTML for static content and JavaScript/Flash for dynamic content. Figure 6.9 shows the structure of a typical XML based gadget document (WSO2 Inc, 2012a). In this structure, every gadget begins with the root element <Module> indicating to the user that this document is a gadget document. The gadget has three main sections:

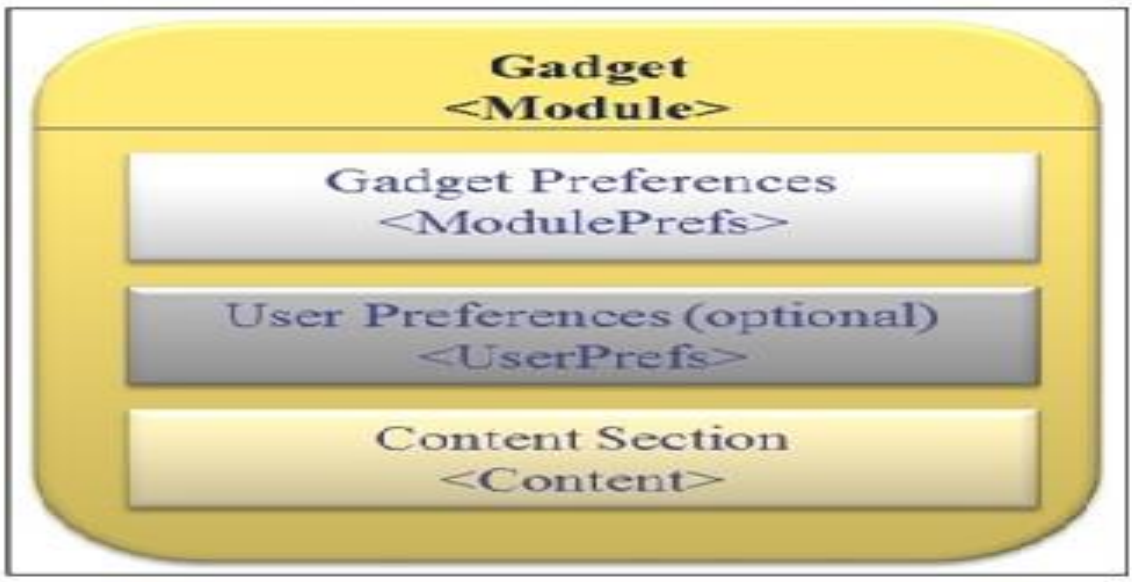


Figure 6.9 Structure of a Google Gadget Document (WSO2 Inc, 2012a).

- **Gadget Preferences:** This is a mandatory section, which starts with the <ModulePrefs> XML element and specifies the main characteristics of a gadget such as name size colour etc.
- **User Preferences:** This is an optional section, which starts with the <UserPrefs> XML element and defines controls that allow users to specify settings for the gadget.
- **Content Section:** This is also a mandatory section which starts with the <Content> XML element and specifies the content of the gadget such as HTML and JavaScript programming logic.

The data from the various sensors needs to be displayed in graphical format on the web. To achieve this, the GG applications need to integrate a web-based visualisation or charting tool. This tool will need to be defined in the content section of the gadget. The charting tool needs to allow the user to develop the charts using JavaScript and HTML.

6.6 Implementation and Testing

This section focuses on the implementation and testing of the proposed µIMP Monitor prototype system by using the developed Google Gadget applications. The µIMP Monitor architecture allows the integration of a number of disparate hardware based platforms and applications. The LVWSM application has already been developed and will be used to collect data from the environment. The machine data will be collected using the Compact-RIO and DAQ applications. Two simple simulation applications have been developed which generate random and counter data which is converted into Restful Web services. The simulation applications will be primarily used for testing the architecture without connecting the actual hardware. These Web Services along with the Web Services created by the LVWSM application will be used to test and evaluate this architecture.

6.6.1 Implementing the µIMP Local Monitor Layer

The µIMP Local Monitor Layer is implemented by hosting the LVWSM application and the Simulator applications. These applications create Restful Web Services for various sensors in the Polymer environment and the simulated counter and random data. The Web Services that are produced by

this layer are used by the µIMP Global Monitor Layer to retrieve the relevant sensor and simulated data depending upon the request from the µIMP Web Layer.

6.6.2 Implementing the µIMP Global Monitor Layer

The WS02 ESB middleware has been implemented to allow the GG web applications to communicate with the backend hardware platforms through the developed applications. The following sections will detail the implementation of the services and artefacts in the µIMP Global Monitor Layer and the communication between the different layers of the architecture.

6.6.2.1 *Implementing the Relational Database and Excel Files*

The data from the various sensors in the Polymer MNT laboratory has to be logged into a relational database as well as in a CSV file format. The LVWSM application creates a CSV file with an excel extension in a separate directory called “C:\wsn\log\”. The data services can access this file by specifying the filename and directory in the XML code.

The data from the LVWSM applications can be logged into two different relational databases namely Microsoft Access and MySQL. In this architecture, a MySQL 5.x database server was configured and setup on a QNAP NAS server with the IP address “143.53.207.64” and port number “3306”. The data services can access the MySQL database by connecting to it using the IP address and port number along with the database name.

6.6.2.2 *Implementing Data Services for Excel Files*

A number of Data Services were created to retrieve data written in excel file format. The LVWSM system logged each sensor node data into excel files. The Data Services were required to retrieve the historical data from each node file and convert and present this data in XML format hence allowing it to be consumed by the ESB. The WSO2 DSS server was used to implement the data services required for retrieving the sensor data from an excel file. The WSO2 DSS server was implemented on a separate PC machine with a unique IP address “143.53.207.228” and port numbers “9443 (for HTTPS) and 9763 (for HTTP)”. The services were developed using the WSO2 Developer Studio and the DSS graphical interface tool. Figure 6.10 shows the format of a data service

```
<data name="MNTService" serviceGroup="Excel">
   <config id="BF50">
      <property name="excel_datasource">./samples/resources/Bfeld2.xls</property>
   </config>
   <query id="voltage" useConfig="BF50">
      <excel>
         <workbookname>Sheet1</workbookname>
         <hasheader>true</hasheader>
         <startingrow>2</startingrow>
         <maxrowcount>-1</maxrowcount>
      </excel>
      <result element="BattenfeldCR" rowName="VReading" defaultNamespace="http://ws.wso2.org/dataservice/uIMP/excelMNTDS/getVoltage">
         <element name="Date" column="Date" xsdType="xs:date" />
         <element name="Time" column="Time" xsdType="xs:time" />
         <element name="Voltage" column="Voltage" xsdType="xs:string" />
      </result>
   </query>
   <query id="temp" useConfig="BF50">
      <excel>
         <workbookname>Sheet1</workbookname>
         <hasheader>true</hasheader>
         <startingrow>2</startingrow>
         <maxrowcount>-1</maxrowcount>
      </excel>
      <result element="BattenfeldCR" rowName="TReading" defaultNamespace="http://ws.wso2.org/dataservice/uIMP/excelMNTDS/getTemperature">
         <element name="Date" column="Date" xsdType="xs:date" />
         <element name="Time" column="Time" xsdType="xs:time" />
         <element name="Temperature" column="Temperature" xsdType="xs:integer" />
      </result>
   </query>
   <query id="humidity" useConfig="BF50">
      <excel>
         <workbookname>Sheet1</workbookname>
         <hasheader>true</hasheader>
         <startingrow>2</startingrow>
         <maxrowcount>-1</maxrowcount>
      </excel>
      <result element="BattenfeldCR" rowName="HReading" defaultNamespace="http://ws.wso2.org/dataservice/uIMP/excelMNTDS/getHumidity">
         <element name="Date" column="Date" xsdType="xs:date" />
         <element name="Time" column="Time" xsdType="xs:time" />
         <element name="Humidity" column="Humidity" xsdType="xs:integer" />
      </result>
   </query>
   <operation name="getVoltage">
      <description>Read Voltage  </description>
      <call-query href="voltage" />
   </operation>
   <operation name="getTemperature">
      <call-query href="temp" />
   </operation>
   <operation name="getHumidity">
      <call-query href="humidity" />
   </operation>
</data>
```

Figure 6.10 Data Service for Retrieving Excel Data.

called “MNTService” created for retrieving data from an Excel file format. The service initialises the connection to the relevant file using the directory and file name in this case “Bfeld2.xls”. The next section specifies the data query which is used to retrieve the data in the excel file. Each query is given a unique name to identify its operation. In this case, the name given to the query used to retrieve the humidity data is “humidity”. The query section uses the excel workbook name “Sheet1” to locate the different elements in the file. A starting point is specified and is used to start reading the data from that position. The output format is specified as XML as well as additional XML tags and namespace to ensure that each element in the excel file is unique and does not conflict with other elements with similar names. The output mappings are specified in the next section. For each column in the excel file an element is mapped according to the data type. The next part of the Data Service exposes

the data queries using web methods or functions. The method created for retrieving the humidity data is called “getHumidity” and this is linked to the “humidity” query, which retrieves the humidity data. The web methods are made available to external users of the Data Service. For each sensor type on the sensor node a data query has been created to retrieve the data from the excel file.

6.6.2.3 *Implementing Data Services for Relational Databases*

A number of Data Services were created on the WSO2 DSS server to retrieve data written to a MySQL relational database. The LVWSM system has the ability to log each sensor node data into MySQL/Access databases. For each node name, a table was created with columns equalling to the number of sensors on each node plus the timestamp column. Data Services were required to retrieve the historical data from each node table in the database and convert and present this data in XML format hence allowing it to be consumed by the ESB. Figure 6.11 shows the part of the Data Service, which sets up the connection to a MySQL relational database called “battendb” .The service, connects to the database residing on a MySQL server using the server IP address and port number and database name. Once connected the next section creates the data query (Figure 6.12) which is used to retrieve the data from the database. The WSO2 DSS allows the use of SQL based data queries. In this output, the data query created is called “LVDBAllQ”. This data query uses the SQL “select” command to select the date/time and all the sensor readings from the table “env11091a”. The next section maps all the columns in the database table that need to be displayed using the SQL query to the relevant data types. The next part of the Data Service exposes the SQL data query (LVDBAllQ) using a web method (data operation) as shown in Figure 6.13. The method created for retrieving all the sensor data from the table is called “getAllReading” and this is linked to the “LVDBAllQ” SQL query which retrieves all the sensor data from the “env11091a” table when the method is called by an external user of the Data Service. For each sensor type on the sensor node a data query has been created to retrieve the data from the MySQL database. The complete XML code for this data service can be found in Appendix E.

```
<data name="LVDBservice" enableBoxcarring="true" serviceGroup="MySQL database">
   <description>A data service to extract data from a MYSQL database which is being updated by a Labview application
   <config id="LVDBSource">
      <property name="org.wso2.ws.dataservice.driver">com.mysql.jdbc.Driver</property>
      <property name="org.wso2.ws.dataservice.protocol">jdbc:mysql://localhost:3306/battendb</property>
      <property name="org.wso2.ws.dataservice.user">root</property>
      <property name="org.wso2.ws.dataservice.password">admin</property>
      <property name="org.wso2.ws.dataservice.xa_datasource_class"></property>
      <property name="org.wso2.ws.dataservice.xa_datasource_properties">
         <property name="URL"></property>
         <property name="User"></property>
         <property name="Password"></property>
```

Figure 6.11 Data Service connection setup for a MySQL database.

```
<query id="LVDBAllQ" useConfig="LVDBSource">
   <sql>select Date_Time, Voltage, Temperature, Humidity from env11091a</sql>
   <result element="sensor_readings" rowName="sensor_reading">
      <element name="Date_Time" column="Date_Time" xsdType="xs:dateTime" />
      <element name="Voltage" column="Voltage" xsdType="xs:double" />
      <element name="Temperature" column="Temperature" xsdType="xs:double" />
      <element name="Humidity" column="Humidity" xsdType="xs:double" />
   </result>
```

Figure 6.12 A Data Service Query for retrieving all sensor data..

```
<operation name="getAllReading">
   <description>Returns all sensor readings from the node with a time stamp </description>
   <call-query href="LVDBAllQ" />
</operation>
<operation name="getVoltageReading">
   <description>Returns the WSN node Voltage Reading from the MYSQL database&#xd; </description>
   <call-query href="LVDBVoltageQ" />
</operation>
<operation name="getTempReading">
   <description>Returns the WSN node Temperature Reading from the MYSQL database      &#xd;   </description>
   <call-query href="LVDBTempQ" />
</operation>
<operation name="getHumidityReading">
   <description>Returns the WSN node Humidity Reading from the MYSQL database  &#xd;   </description>
   <call-query href="LVDBHumidityQ" />
</operation>
```

Figure 6.13 Web methods created for exposing the data queries.

6.6.2.4 *Data Service Web Services Description Language (WSDL) File Creation*

A WSDL file is a Web Service interface description language specified using XML. It is used to describe the functionality of a Web Service. A WSDL file provides a description of how the Web Service is to be called, the parameters it expects, and the data structures it will return. The WSDL specification (W3C, 2007b) provides a specific XML format for documents used to describe web services. The WSDL exposes the service to the outside world as a collection of network ports and endpoints. Abstract definitions of the ports and messages allow the definitions to be reused. A port is defined by linking an IP address with the reusable port bindings. The data being exchanged is described using abstract descriptions and the port types give the abstract collections of supported operations. The WSDL is used in conjunction with SOAP and XML schema to create Web services. Clients that connect to this Web Service can

determine what operations are available by reading the WSDL file. These operations can be called by the client using SOAP through the use of XML and HTTP. Figure 6.14 shows an overview of the WSDL file created for the "LVDBService" Data Service used for retrieving data from the MySQL database. In this overview a single operation (web method) called "getAllreading" has been shown which is used to extract all the sensors data from the MySQL database. A single WSDL port type and binding has been shown in this output. The SOAP11 endpoint is shown along with the location address.

```
− <wsdl:definitions targetNamespace="http://ws.wso2.org/dataservice">
  − <wsdl:documentation>
      A data service to extract data from a MYSQL database which is being updated by a Labview application with WSN node data in RT.
    </wsdl:documentation>
  + <wsdl:types></wsdl:types>
    <wsdl:message name="getAllReadingRequest"/>
  − <wsdl:message name="getAllReadingResponse">
      <wsdl:part name="parameters" element="ns0:sensor_readings"/>
    </wsdl:message>
  − <wsdl:portType name="LVDBservicePortType">
  − <wsdl:operation name="getAllReading">
    + <wsdl:documentation></wsdl:documentation>
      <wsdl:input message="ns0:getAllReadingRequest" wsaw:Action="urn:getAllReading"/>
      <wsdl:output message="ns0:getAllReadingResponse" wsaw:Action="urn:getAllReadingResponse"/>
      <wsdl:fault message="ns0:DataServiceFault" name="DataServiceFault" wsaw:Action="urn:getAllReadingDataServiceFault"/>
    </wsdl:operation>
  − <wsdl:binding name="LVDBserviceSOAP11Binding" type="ns0:LVDBservicePortType">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
      <wsdl:operation name="getAllReading">
        <soap:operation soapAction="urn:getAllReading" style="document"/>
      − <wsdl:input>
          <soap:body use="literal"/>
        </wsdl:input>
      − <wsdl:output>
          <soap:body use="literal"/>
        </wsdl:output>
      − <wsdl:fault name="DataServiceFault">
          <soap:fault use="literal" name="DataServiceFault"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:service name="LVDBservice">
      − <wsdl:port name="SOAP11Endpoint" binding="ns0:LVDBserviceSOAP11Binding">
          <soap:address location="http://192.168.0.2:9763/services/LVDBservice.SOAP11Endpoint/"/>
        </wsdl:port>
      − <wsdl:port name="SecureSOAP11Endpoint" binding="ns0:LVDBserviceSOAP11Binding">
          <soap:address location="https://192.168.0.2:9443/services/LVDBservice.SecureSOAP11Endpoint/"/>
        </wsdl:port>
```


Figure 6.14 An overview of the "LVDBService" Data Service WSDL file.

6.6.2.5 *Implementing ESB Proxy Services*

In the µIMP Global Monitor Layer the ESB based middleware server consumes the data services from the DSS server as well as the Restful Web Services from the process monitoring applications. The middleware component of this layer was implemented using the WSO2 ESB. The WSO2 ESB server was setup on a separate PC machine with a unique IP address "143.53.207.228" and port numbers "9445 (HTTPS), 9765 (HTTP)". The ESB is also available using the

```
<proxy xmlns="http://ws.apache.org/ns/synapse" name="ExgetHumidP" transports="https,http" statistics="disable" trace="disable" startOnLoad="true">
   <target endpoint="ExgetHumidEP">
      <outSequence>
         <send />
         <log separator="," />
      </outSequence>
   </target>
</proxy>
```

Figure 6.15 Customized Proxy Service for retrieving Excel file data.

HTTP-NIO non-blocking transport for reliability and can be accessed on the port numbers “8280 (HTTP) 8243 (HTTPS). A number of proxy services have been created for each of the external services and the ESB uses these services to receive and process the messages if necessary before sending them off to the actual services using endpoints. The different types of proxy services that have been created in the µIMP Global Monitor Layer include Pass Through, Logging, WSDL Based, and Customised services. Figure 6.15 shows

```
<proxy xmlns="http://ws.apache.org/ns/synapse" name="LVDBPservice" transports="https,http" statistics="disable" trace="disable" startOnLoad="true">
   <target>
      <inSequence>
         <log level="custom" separator=",">
            <property name="MESSAGE" value="Sending Request to LVDBP Service...." />
         </log>
         <send>
            <endpoint key="LVDBserviceEP" />
         </send>
      </inSequence>
      <outSequence>
         <log level="custom" separator=",">
            <property name="MESSAGE" value="Sending the Response back to the client from LVDBP Service....." />
         </log>
         <send />
      </outSequence>
      <faultSequence>
         <makefault version="soap12">
            <code xmlns:soap12Env="http://www.w3.org/2003/05/soap-envelope" value="soap12Env:Receiver" />
            <reason expression="get-property('ERROR_MESSAGE')" />
            <node />
            <role />
            <detail>fault_detail</detail>
         </makefault>
         <header name="To" action="remove" />
         <send />
      </faultSequence>
   </target>
   <publishWSDL>

   This section includes the complete WSDL file for the data service called LVDBService.

   </publishWSDL>
</proxy>
```

Figure 6.16 Customized Proxy Service using the WSDL file of the actual Data Service for retrieving MySQL data.

a simple customized proxy service called “ExgetHumidP” which has been created to send a request from a GG client application to the “MNTService” Data Service running on the DSS server. The “MNTService” retrieves sensor data stored in an Excel file. Whereas Figure 6.16 shows an overview of a customized proxy service called “LVDBPservice” which has been created to send a request from a Google Gadget client application to the “LVDBservice” Data service running on the DSS server. The “LVDBservice” retrieves sensor data from the MySQL database using the different methods defined in the actual Web Service. This“LVDBPservice” proxy service uses the WSDL file of the “LVDBservice” data service to access the methods defined in this actual data service. An overview of one of the methods can be found in section 6.6.2.2.

6.6.2.6 *Implementing ESB Endpoints*

The ESB exposes service endpoints that accept messages from clients. These services act as proxies for external services and the ESB mediates the messages before they are proxied to the actual service. Figure 6.17 shows an example of the use of endpoints in which the LVWSNPROXY is exposed through an endpoint and acts as a proxy for the services it is going to call. In the case of the LVWSNPROXY it was created as a proxy service for the “LVWSNService” created by the LVWSM application. A proxy service maybe a SOAP or REST/Plain Old XML (POX) service over HTTP/S or SOAP, POX, Plain Text or Binary / Legacy service for other transports such as JMS and VFS file systems.

For each proxy service, an endpoint was created which holds the destination address of the actual service. Figure 6.18 shows an endpoint called “ExgetHumidEP” used by the “ExgetHumidP” proxy service. The external destination address for the message sent by the GG client application is “https://localhost:9443/services/MNTService/getHumidity”. This address uses the IP address of the DSS server and the name of the data service along with the method, which is used to retrieve the data from the Excel file. Whereas Figure 6.19 shows an endpoint called “LVDBserviceEP” used by the “LVDBPservice” proxy service. The external destination address for the message sent by the Google Gadget application is

"https://localhost:9443/services/LVDBservice". This address is similar to the one used for the “ExgetHumidEP” as it has the IP address along with the name of the actual data service. The only difference being is that the name of any particular method is not used. This is not required since the WSDL file of the actual data service is being used from which all the methods defined for this service can be accessed.

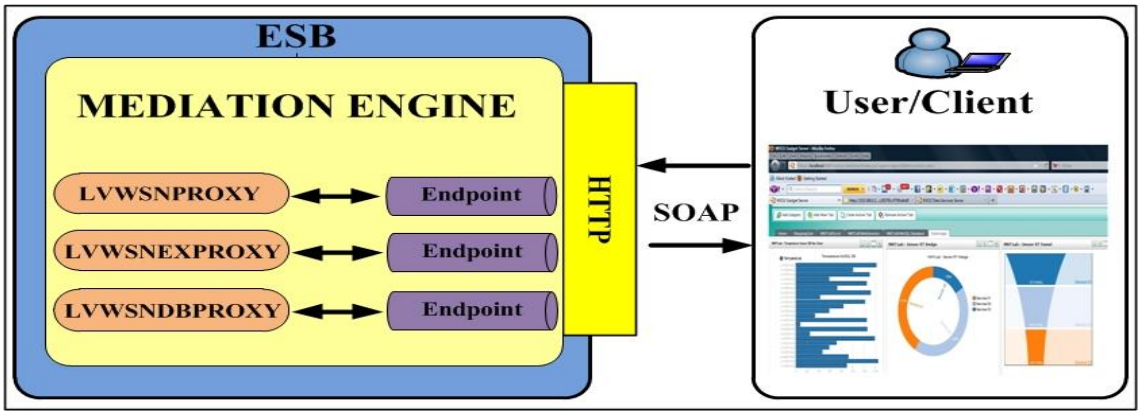


Figure 6.17 Proxy Service exposed using endpoints.

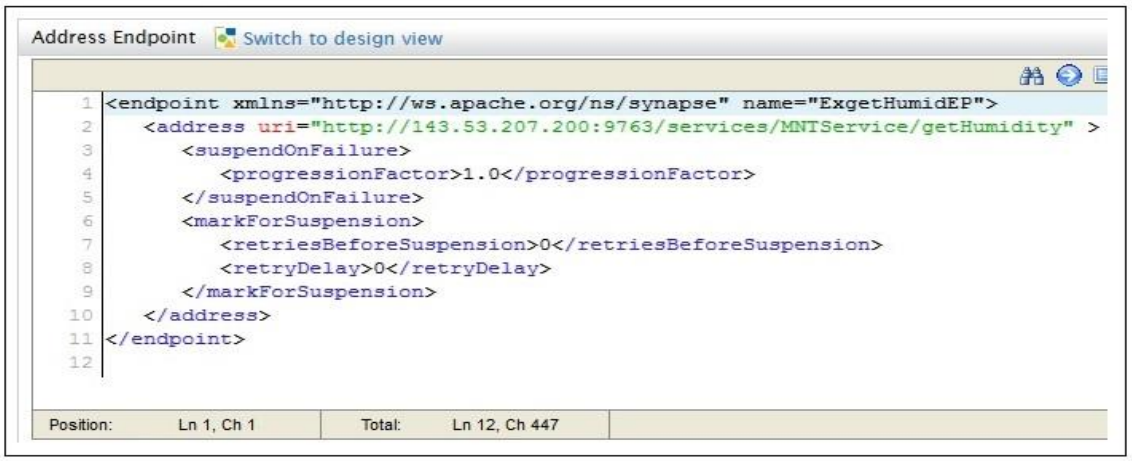


Figure 6.18 Endpoint created for the "ExgetHumidP" proxy service.

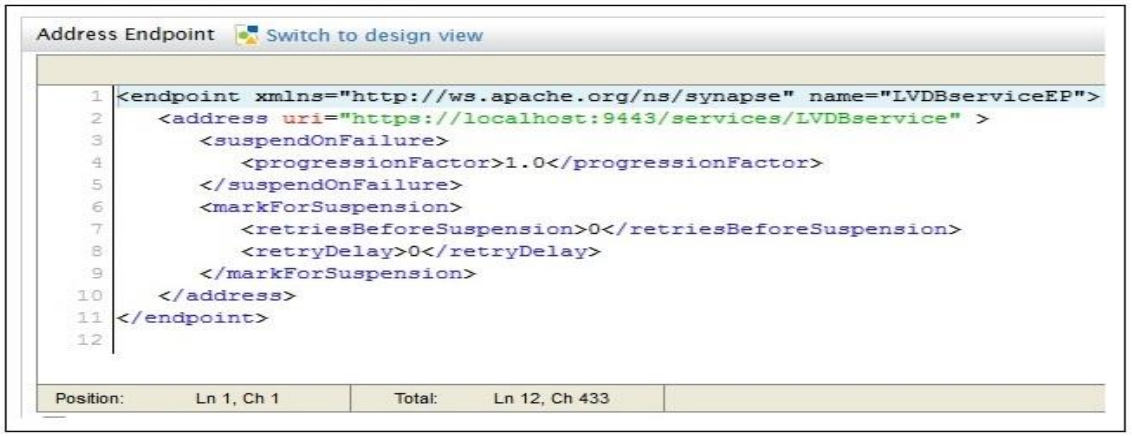


Figure 6.19 Endpoint created for the "LVDBservice" proxy service..

6.6.2.7 *Use of ESB Message Mediators*

For each proxy service, a mediator of one form or the other was used to transform or change the incoming and outgoing messages. Figure 6.20 shows the mediators used in the LVDBPservice proxy service.

```
In Sequence Mediators
<inSequence xmlns="http://ws.apache.org/ns/synapse">
   <log level="custom" separator=",">
      <property name="MESSAGE" value="Sending Request to LVDBP Service...." />
   </log>
   <send>
      <endpoint key="LVDBserviceEP" />
   </send>
</inSequence>

 Out Sequence Mediators
<outSequence xmlns="http://ws.apache.org/ns/synapse">
   <log level="custom" separator=",">
      <property name="MESSAGE" value="Sending the Response back to the client from LVDBP Service....." />
   </log>
   <send />
</outSequence>
```

Figure 6.20 Sequence mediators used for incoming and outgoing messages.

6.6.2.8 *Creation of ESB WSDL Files*

As described in section 6.6.2.4 a service needs to have a WSDL file, which describes the functionality and exposes the service to the outside world as a collection of network ports and endpoints. External clients connect to the proxy service by using the information in its WSDL file. For each proxy service created in the µIMP Global Monitor Layer a WSDL file was created.

6.6.3 Implementing the µIMP Web Layer

The µIMP Web Layer is implemented on the WSO2 Gadget Server (GS). The WSO2 GS is built upon the WSO2 Carbon platform and uses the GGs API to create and deploy GGs. A number of GG applications have been developed to monitor the various sensors data in the µIM process. The GG Server allows the creation of tabs to allow the grouping of similar types of gadgets. In this project, the µIM process life cycle depicted in Figure 6.4 has been modelled using each Tab to represent each process stage in the life cycle. Figure 6.21 shows the created tabs using the µIM process life cycle. In each tab, a number of gadgets were created to depict the sensor data in each of the stage processes. Figure 6.22 shows the start of a gadget created to display data LVSimulator Counter simulation data. As it can be seen, the gadget starts with the XML <Module> element. It has a Module Preference section gives the name and title of the author and gadget along other features such as having the gadget to change its height size in runtime. Figure 6.23 shows the content section that defines the

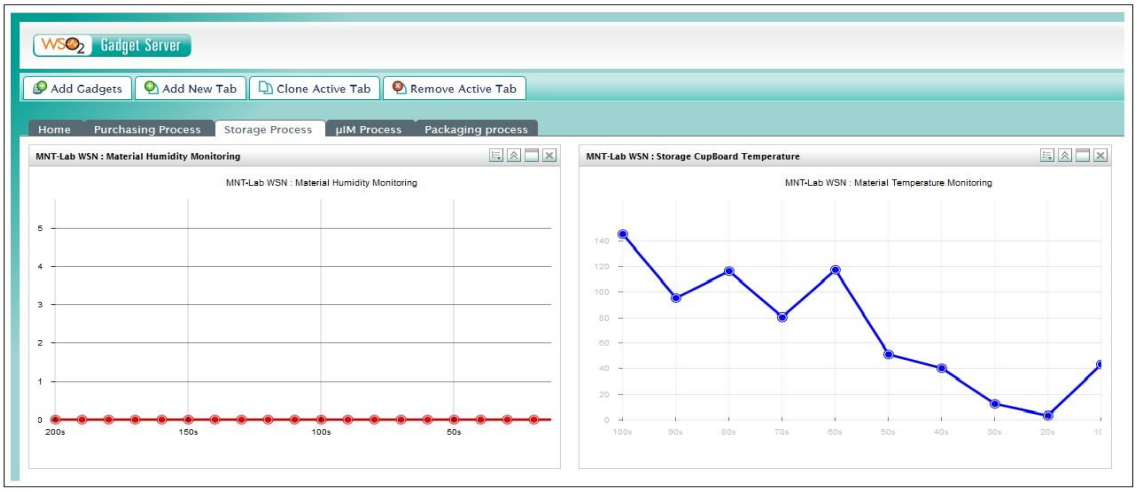


Figure 6.21 Gadget Portal used to organize gadgets.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
    <ModulePrefs
            title="MNT-Lab LVSIM : Labview Counter Generator"
            title_url="www.bradford.ac.uk"
            author="Umar Raza"
            author_email="r.umar@bradford.ac.uk" scrolling="true" height="350">
        <Require feature="dynamic-height" />
    </ModulePrefs>
```

Figure 6.22 The start of a GG developed to display LVSimulator counter simulation data.

```
<Content type="html" view="canvas"><![CDATA[
 <html>
 <head>
   <script type="text/javascript" src="js/ViskitG.js"></script>

      <script language="javascript" type="text/javascript">

          var dataProvider = new Viskit.p.ProviderGETMakeRequest("http://localhost:8282/services/LVCounterProxy");
          var dataFilter = new Viskit.f.BasicFilter(["Response", "Terminal"], ["Name"],["SValue"]);
          var timer = new Viskit.u.Timer(1000);

          function globalTick() {
              dataProvider.pullData();
          }
          function lineTooltip(data) {
              return data["SValue"];
          }
          function lineOnClick(data) {
              alert("clicked: " + data["SValue"]);
          }
          window.onload = function () {

              var lineChart = new Viskit.s.chart.protovis.LineChart("chart","MNT-Lab LVSim : Counter Simulation","");
```

Figure 6.23 Content section of a GG.

type of content to be used in this gadget. In this output JavaScript and HTML was used. Within the contents section of the gadgets created for this project, a visualization tool from Stanford University called “VisKit” (WSO2 Inc, 2012c) is utilized. This tool has been further developed by the WSO2 team and allows the creation of dynamic visualization charts using JavaScript and HTML. The Viskit tool was used to develop a dynamic line chart for displaying the counter data.

From the code, it can be seen that the Viskit library uses the XMLHttpRequest DOM API in JavaScript to send a HTTP request to the LVCounterProxy service running on the µIMP Global Monitor Layer ESB.

6.6.4 The µIMP Monitor Test bed Layout

The µIMP monitor project has been implemented in the centre for Polymer MNT at the University of Bradford. A layout of the test bed is shown in Figure 6.24. The Polymer MNT centre has an extensive range of processing hardware specifically designed for micro and Nano moulding activities. The Jennic wireless sensor nodes were deployed with on-board (temperature, humidity, voltage) to monitor the environment in the MNT centre including the clean rooms, material storage cupboard and the ambient environment. A NI CDAQ device, which has 32 analogue inputs and 2 digital inputs, is installed in one of the µIMP machine to monitor the various process temperature and pressure values. While a NI Compact RIO embedded controller with 32 analogue inputs was installed in another µIM machine to monitor temperature, pressure and the piston displacement and velocity. The LVWSM and the LVSimulator applications were setup on a separate PC machine. The WS02 ESB and Gadget Server instances have been setup on a single PC machine; while a separate PC machine hosts the data service server, which connects to a MySQL database residing on a QNAP NAS server.

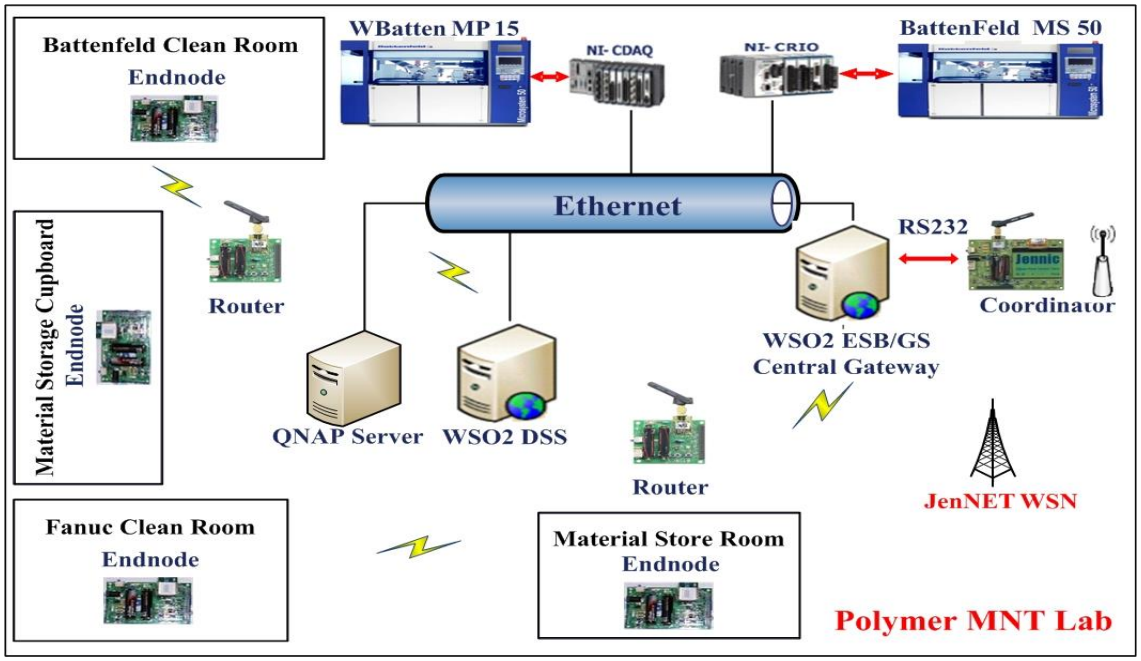


Figure 6.24 Polymer MNT Laboratory Test bed Layout.

6.6.5 µIMP Local Monitor Layer to µIMP Web Layer Communication.

This section gives the details of the communication pattern between the µIMP Local Monitor Layer and the µIMP Web Layer when testing the complete architecture. Once the gadget has been designed, it is then uploaded into the gadget web portal on the GG Server. The web portal allows the gadgets to be organized as part of the different stages of the µIM process lifecycle as described earlier in section 6.6.3 using tabs. Each gadget in the portal follows the message communication pattern shown in Figure 6.25. The GG applications send SOAP requests using HTTP transport to the ESB server. The ESB Server then formats and converts this message to allow it to be sent to the Restful Web services. The internal mechanism used by the ESB to send and receive messages is explained in more detail in the next section. The ESB uses Plain Old XML (POX)/Restful over HTTP/HTTPS transport to forward to this message to the LVWSM Local application. The LVWSM sends the Restful response back to the ESB which converts the message back to SOAP format and sends the response back to the GG application using SOAP over HTTP transport. In this communication pattern, depending upon the type of data required (historical or the latest data in in real-time) the gadgets running in the web portal send requests to the relevant proxy services residing on the ESB server. If the request is for historical data, then the ESB routes the request to the DSS server

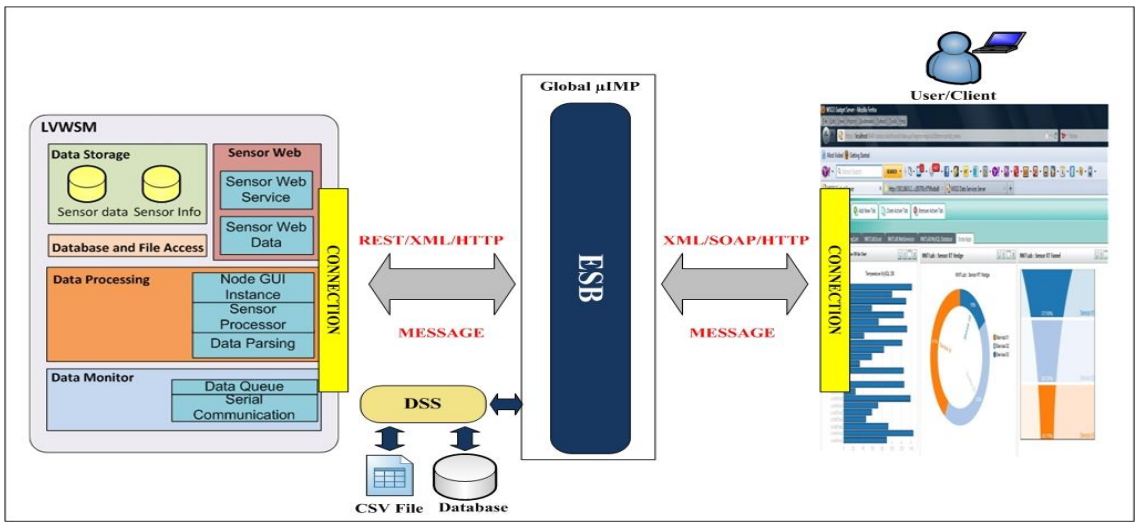


Figure 6.25 Message communication pattern between different layers.

which hosts the data services for retrieving historical data stored in Excel and MySQL database. If the request is for current data in almost real-time on the web then the ESB routes the request direct to the relevant Restful Web

Services running in the Applications in the µIMP Local Monitor Layer. These Web services then retrieve the real-time data from using the hardware in the µIMP Hardware Layer.

6.6.5.1 *Local Monitor Layer to µIMP Web Layer Message Exchange*

This section details the message exchange between the µIMP Local Monitor Layer and the µIMP Web Layer via the µIMP Global Monitor Layer. The data from the µIMP Local Monitor Layer applications is logged into a file and a MySQL database as well as being converted into a Restful Web Service. The WSO2 Data Service Server (DSS) in the µIMP Global Monitor Layer creates data services to manage the historical sensor data stored in the file and database. The Local Monitor Layer application LVWSM creates Restful Web Services for the sensor nodes on-board three sensors (temperature, humidity, voltage) whilst the LVSimulator creates Restful Web Services for the counter and random data. For each type of data, a Google Gadget is created in the µIMP Web Layer. The µIMP Global Monitor Layer creates a proxy service for each Restful Web Service. The µIMP Global Monitor Layer receives messages by using the message relay streaming mechanism described in section 6.4.2.1. The message builder component parses the messages arriving via different transports. Depending on the actual content type of the incoming message, the ESB selects a suitable message builder, which will then parse the message content and convert it into the Apache AXIOM based XML infoset. In Figure 6.26 the message builder uses SOAP message builder to parse the SOAP message for the LVWSNPROXY service, which requests data from the Restful Web Services created by the LVWSM application. The message goes through the ESB mediation engine, which performs any mediation, specified in the proxy service such as data transformation and augmentation and gets passed onto to an endpoint to be routed to the relevant address. The message formatter then converts the message to be routed to the Restful Web service. A message formatter, which is the opposite of the builder, converts the message back to its original format by referring the content type just before handing it over to the transports again for routing. The reverse happens when the Restful Web service returns the data from the sensors. A Restful builder is used on the return path. The message again goes through the ESB for any mediation and

then is passed to an endpoint at which point a message formatter is used once again to convert the message back to SOAP format and sent to the GG application.

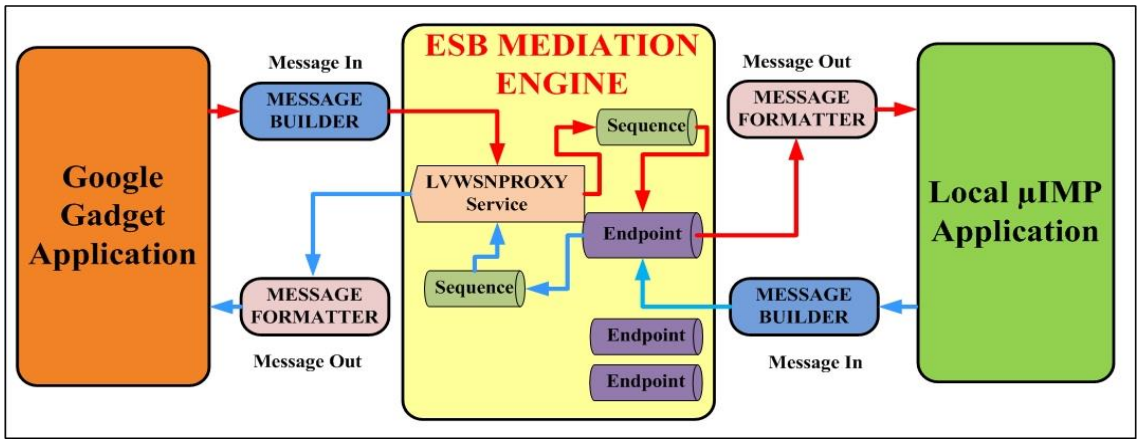


Figure 6.26 ESB message mediation using message Builders and Formatters.

6.6.6 Communication Validation between the Layers in the Architecture

This section details the message communication validation between the different layers of the architecture. Figure 6.27 highlights five points of communication points, which need to be validated to ensure that the proposed architecture is working as expected. The five points of communication that have been identified and which need to be validated are:

- Web layer to ESB Communication
- ESB to DSS Communication
- ESB to Local Application Communications
- DSS to Database Communication
- DSS to Excel File Communication

These five points of communication need to be tested and validated individually before integrating the three layers in the architecture. The testing for these layers was carried out by using the WSO2 TryIt tool (WSO2 Inc, 2012f). The TryIt tool provides a quick and easy way to test Web Services WSDL file. The tool provides a mechanism of testing a WSDL by creating endpoints, which use different transports on the go. The created Web Services can be tested by using the services WSDL file. Using the WSDL file the WSO2 TryIt tool allows the Web Service to be invoked using the operations specified in the WSDL file. For more information on the TryIt tool, see Appendix E.2.

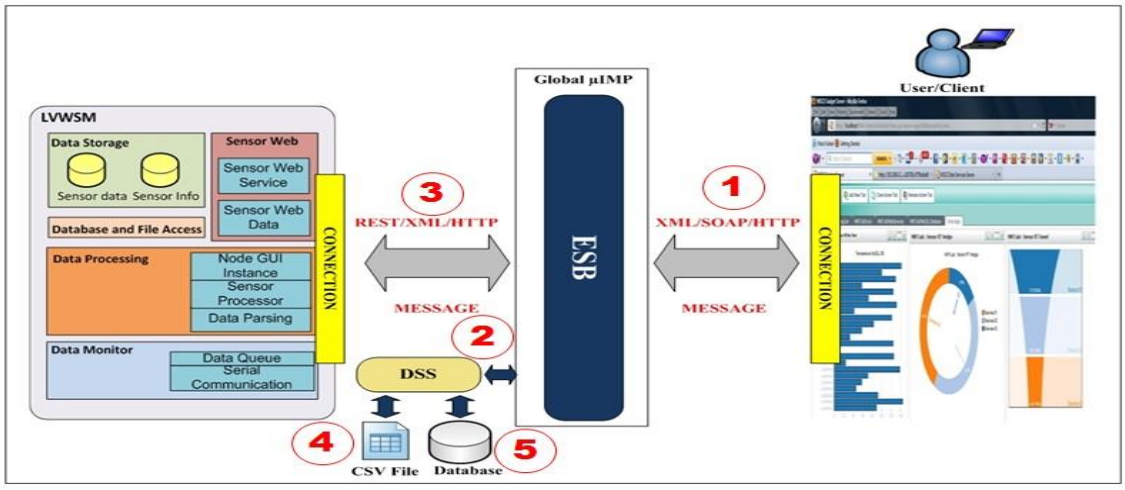


Figure 6.27 Communication points to validate.

6.6.6.1 *Web layer to ESB communication*

When a client web application in the µIMP Web layer such as a GG requests data from the backend hardware in the µIMP Hardware Layer, it first sends a request to the ESB in the µIMP Global Monitor Layer. The GG application uses JavaScript and XML/HTML to communicate with GG API. The XMLHttpRequest DOM API in JavaScript is used to send a HTTP request for the process data gathered from the various hardware platforms. An open method inside the main body of a GG application is used to initialise the XMLHttpRequest. For example

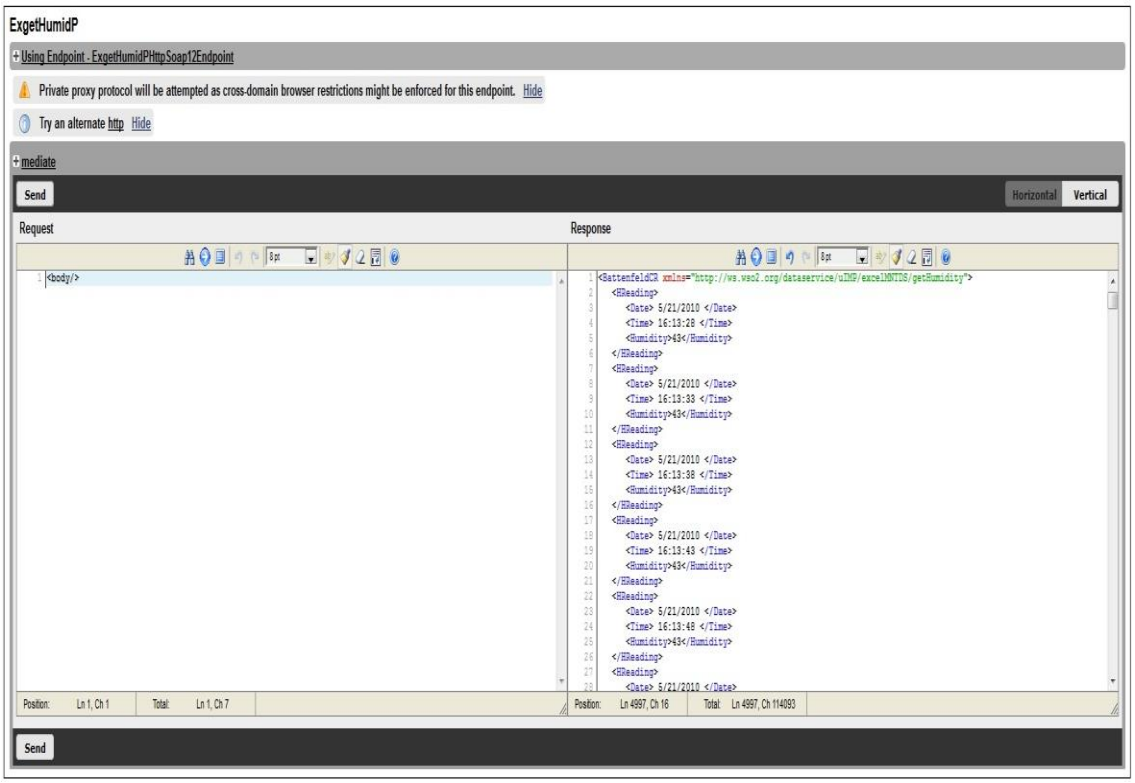


Figure 6.28 Output from the WSO2 TryIt tool for "ExgetHumidP" proxy service.

the URL to invoke the proxy service for the Humidity readings stored in an Excel file is (http://143.53.207.228:9445/services/ExgetHumidP). A successful invocation of the open method sends a SOAP message to the µIMP Global Monitor Layer ESB to invoke the proxy service which in turn invokes the Restful Web Service. A successful call to the send method returns data in a DOM document object. To validate that the communication to the ESB proxy services is working, the "Try this Service" link next to the ExgetHumidP service deployed on the ESB is used. Once this link is clicked the WSO2 TryIt tool takes over and uses the WSDL file for this proxy service to make available any operations for this service. Figure 6.28 shows the resultant window when the TryIt tool is used to test the proxy service. When the send button is clicked, the proxy service is invoked using the only default operation (mediate) the "ExgetHumidP" proxy service has. The response from this invocation can be seen in the window on the right which displays the humidity data read from the excel file in XML format. From this response it can be verified that the proxy service has been successfully invoked by the web client (in this case it is the TryIt tool) and that the communication between the µIMP Web Layer and µIMP Global Monitor Layer ESB is working as expected.

6.6.6.2 *ESB to DSS Communication*

The ESB to DSS communication has been partially verified in the previous section as the "ExgetHumidP" proxy service on the ESB sent a request to the"MNTService" data service on the DSS server which responded back with the humidity data from the excel file. To complete the verification that the DSS server is responding as expected the TryIt tool is used to directly invoke the "MNTService" data service on the DSS server. Figure 6.29 shows the results from this invocation and it can be seen that the "MNTService" has three operations to extract data from the different sensors. When the data service is invoked using the "getHumidity" operation, the response is shown in the window on the right with the humidity data being displayed in XML format. From this response, it can be verified that the TryIt tool using the various operations available has successfully invoked the data service. It can also be verified that the communication between the µIMP Global Monitor Layer ESB and the DSS is working as expected.

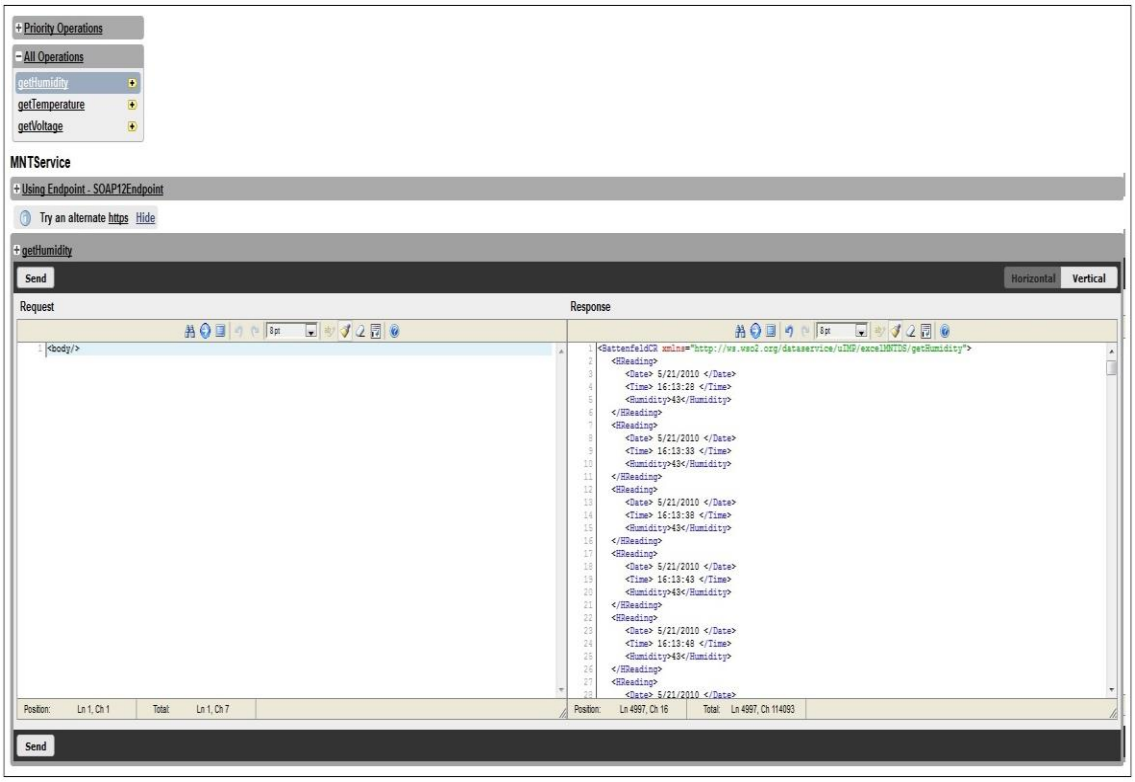


Figure 6.29 Invocation of the "MNTservice" data service using the WSO2 TryIt tool.

6.6.6.3 *ESB to Local Application Communication*

Once the ESB receives a request from the web applications for data, it determines the recipient of the request in the µIMP Local Monitor Layer and sends a request to the Restful Web services in this Layer. To validate and test the communication between the µIMP Global Monitor Layer ESB and the µIMP

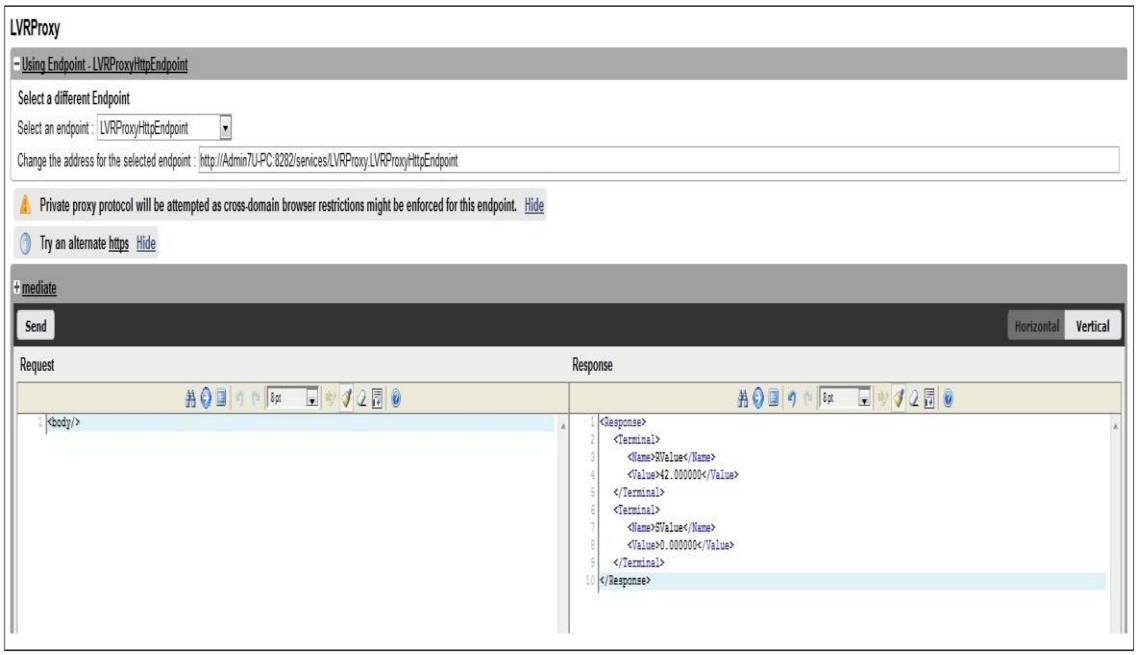


Figure 6.30 Invocation of a Restful Web Service using the TryIt tool.

Local Monitor layer applications is working; the TryIt tool was used to invoke a Restful Web Service through a proxy service from the ESB. Figure 6.30 shows the invocation of a Restful Web Service called “random” via the “LVRProxy”. From this invocation, it can be seen that the µIMP Global Monitor Layer ESB has successfully invoked the actual Restful Web Service “random” which retrieves the random data generated by the LVsimulator application. From the response in the right hand side window, it can be verified that the communication between the µIMP Global Monitor Layer ESB and the µIMP Local Monitor Layer applications is working as expected.

6.6.6.4 *DSS to Database Communication*

The DSS in the µIMP Global Monitor Layer is responsible for creating data services, which can retrieve data from various sources. The DSS server needs to extract data from a MySQL database, which holds the data for the WSN sensors. To test the communication between the DSS Server and the MySQL database the TryIt tool was used to directly invoke a data service that retrieves data from the MySQL database. Figure 6.31 shows the “LVDBservice” that was

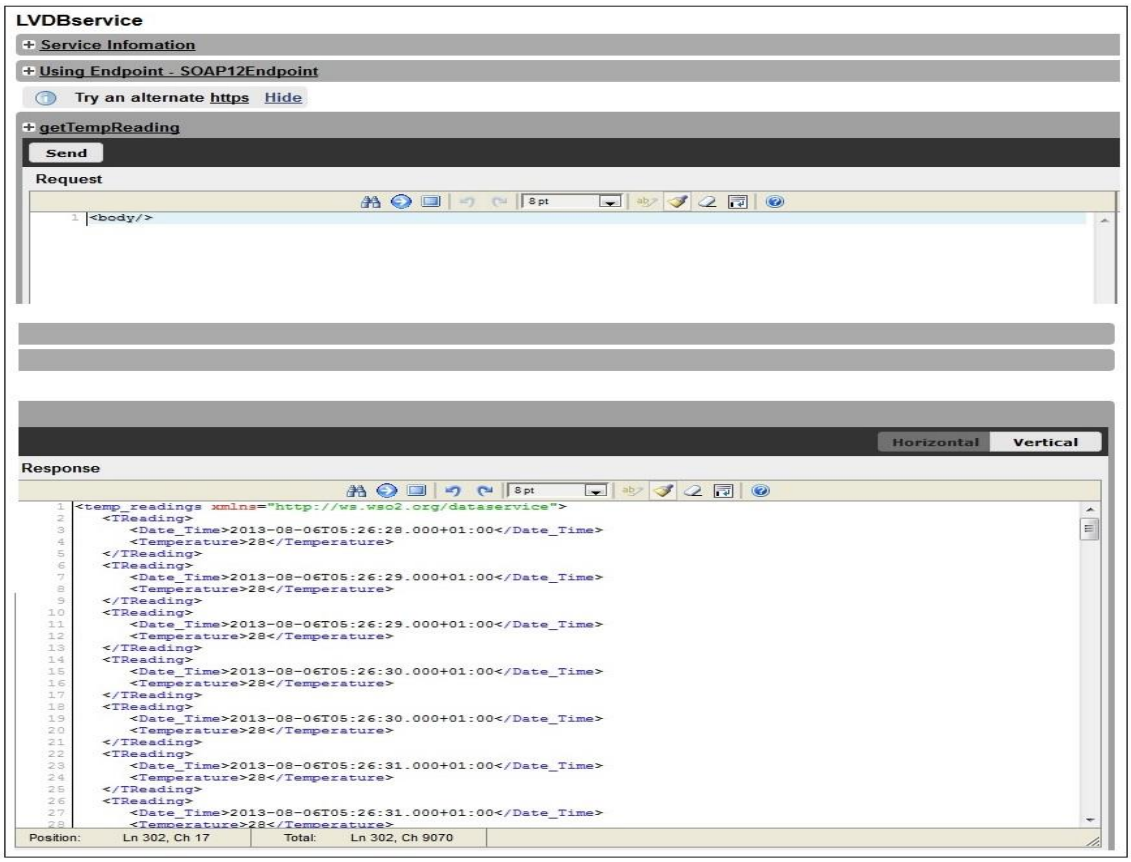


Figure 6.31 Direct invocation of a data service for retrieving MySQL data.

invoked using the “getTemperature” method by the TryIt tool. The response in the bottom window from the service shows the retrieved temperature data from the MySQL database. From this response, it can be verified that the communication between the DSS server in the µIMP Global Monitor Layer and the MySQL database is working as expected.

6.6.6.5 *DSS to Excel File Communication*

The DSS in the µIMP Global Monitor Layer is responsible for creating data services, which can retrieve data from various sources. The DSS server needs to extract data from an Excel file, which holds the data for the WSN sensors. To test the communication between the DSS Server and the Excel file the TryIt tool was used to directly invoke a data service that retrieves data from an Excel file. Figure 6.32 shows the “MNTService” that was invoked by the TryIt tool. The response in the bottom window from the service shows the retrieved data from the Excel file. From this response, it can be verified that the communication between the DSS server in the µIMP Global Monitor Layer and the Excel files is working as expected.

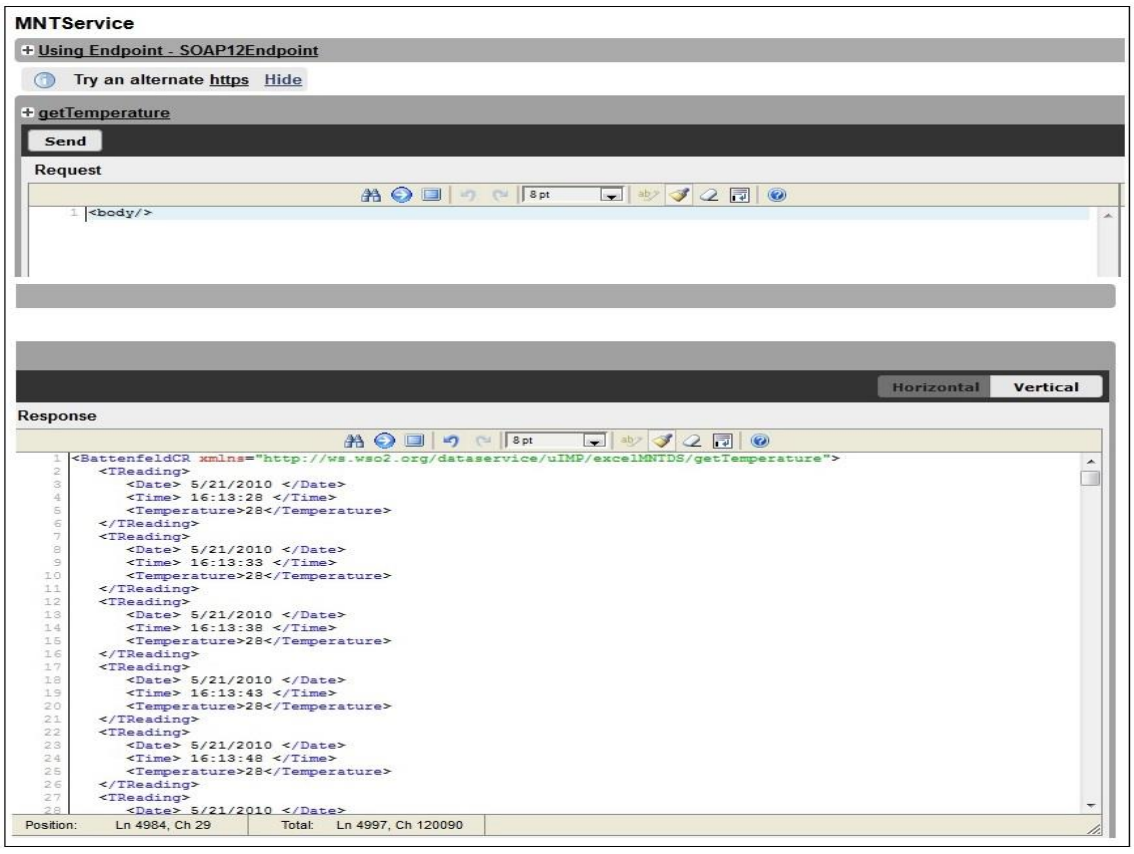


Figure 6.32 Direct invocation of a data service for retrieving Excel file data.

6.6.7 Testing

The complete architecture was first tested by using simulated data and then replacing this with the LVWSM application.

6.6.7.1 *Testing using the LabVIEW Simulator Data*

First, the integration of the proposed test bed environment was demonstrated by using the LVSimulator applications written in LabVIEW which generated random number and counter data. The random number application generated a random value and converted it into a Web Service every second. The counter application counted from zero up to a user specified value and then back to zero again continuously. The counter data was also converted into a Web service. Both the counter and random simulated data were converted into a Restful Web Service called "uIMPSimpleWS". The output format for this Restful Web Service was in XML as shown in Figure 6.33. The WSO2 GG Server was used to visualize the data as gadgets. On the µIMP Global Monitor Layer ESB a proxy service (Figure 6.34) was created and exposed through an endpoint pointing to the Restful Web Services. On the WSO2 GG server three gadgets

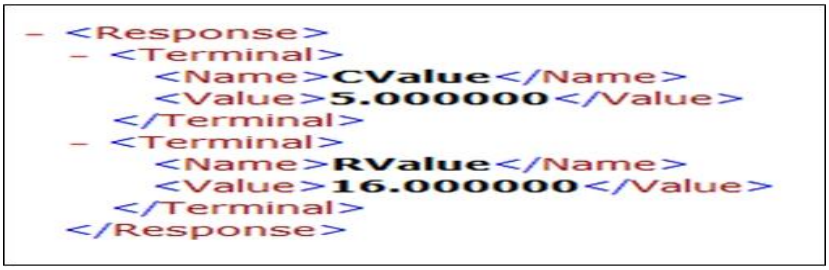


Figure 6.33 XML output from a Restful Web Service.

Figure 6.34 "LVSimpleProxy" exposed through an endpoint.

were developed - one gadget each for the counter and random data and third gadget to display both results on the same chart (Figure 6.35). The gadgets

send XML requests to the µIMP Global Monitor Layer ESB at regular intervals and visualize the result if an update is available. A number of different gadget types were tested and Figure 6.36 shows the same data being displayed using a different gadget types along with a welcome gadget for the Polymer MNT Laboratory.

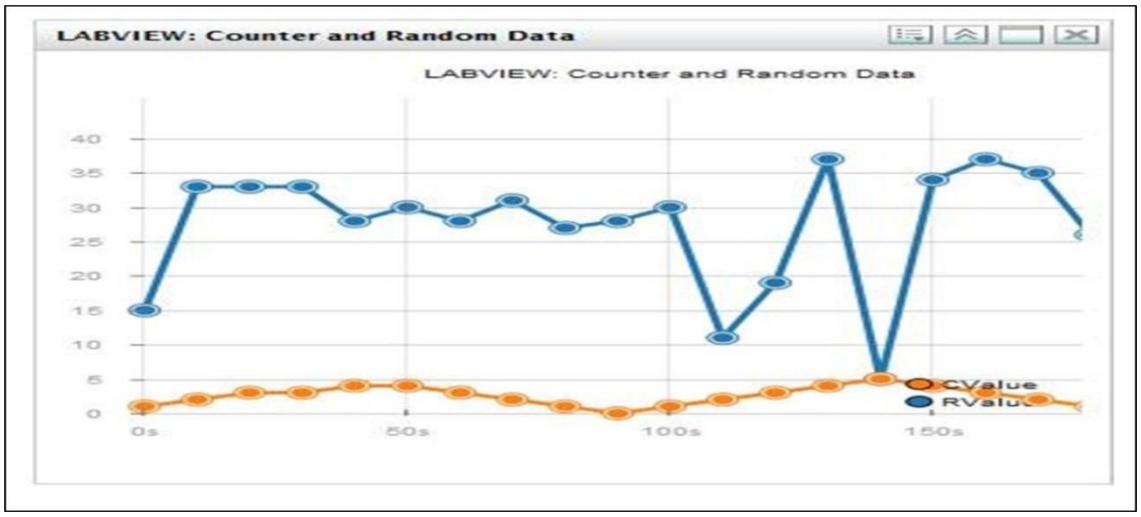


Figure 6.35 A single gadget displaying both counter and random data.

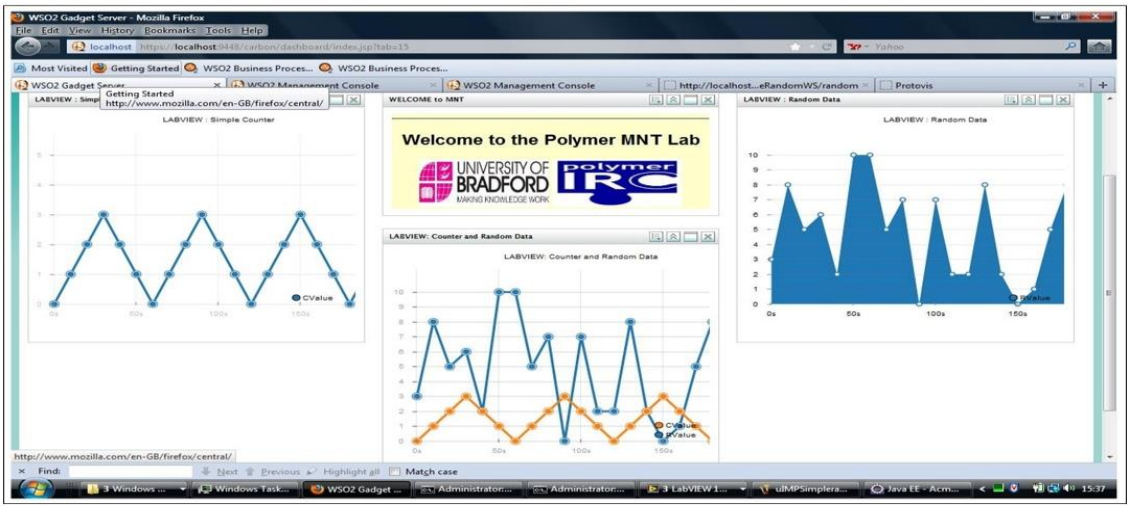


Figure 6.36 Different types of Gadgets used to display the simulated sensor data.

6.6.7.2 *Testing using the LVWSM system*

In the second test the simulated data was replaced with Jennic WSN data, the sensor nodes in the Jennic WSN have integrated Sensirion SHT11 single-chip multi-sensor module for the measurement of temperature and relative humidity. The two sensor readings and node voltage reading were extracted and converted into a RESTful web service. On the µIMP Global Monitor Layer ESB, proxy services were created and exposed through endpoints. A number of

gadgets were created to visualize the results in different formats depending upon the type of data required (historical or real-time). The real-time data from the μIMP Local Monitor Layer LVWSM application was displayed using line charts as shown in Figure 6.37. This figure shows two gadgets displaying humidity and temperature data, which is being updated when the sensor node was used to monitor the environment. On the other hand, the historical data from an Excel file is displayed in three separate gadgets using bar charts as shown in Figure 6.38.

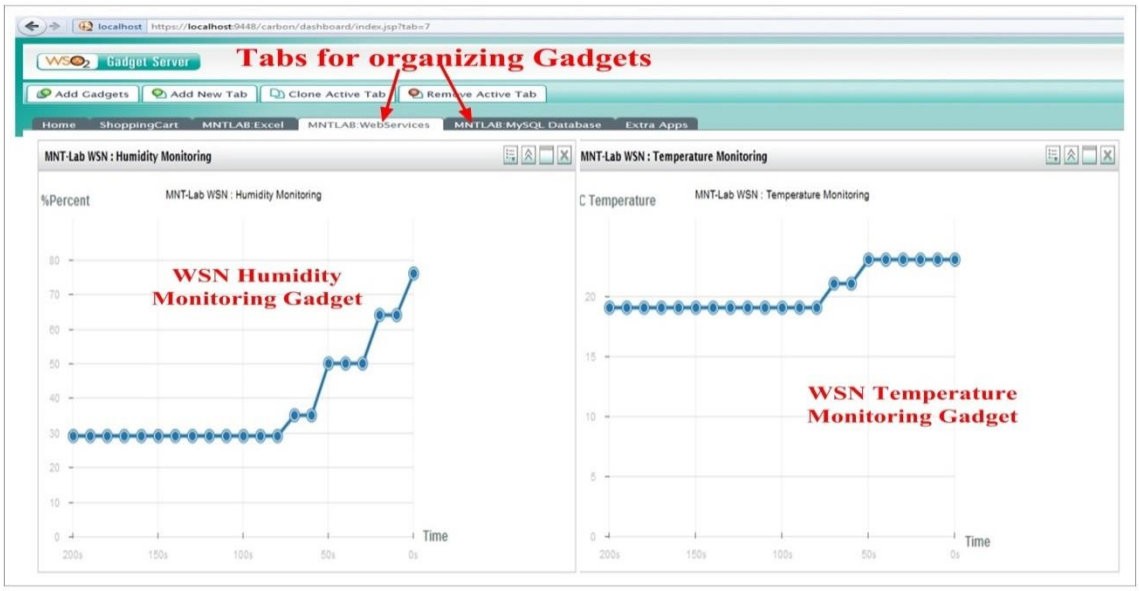


Figure 6.37 WSN Humidity and Temperature Gadgets.

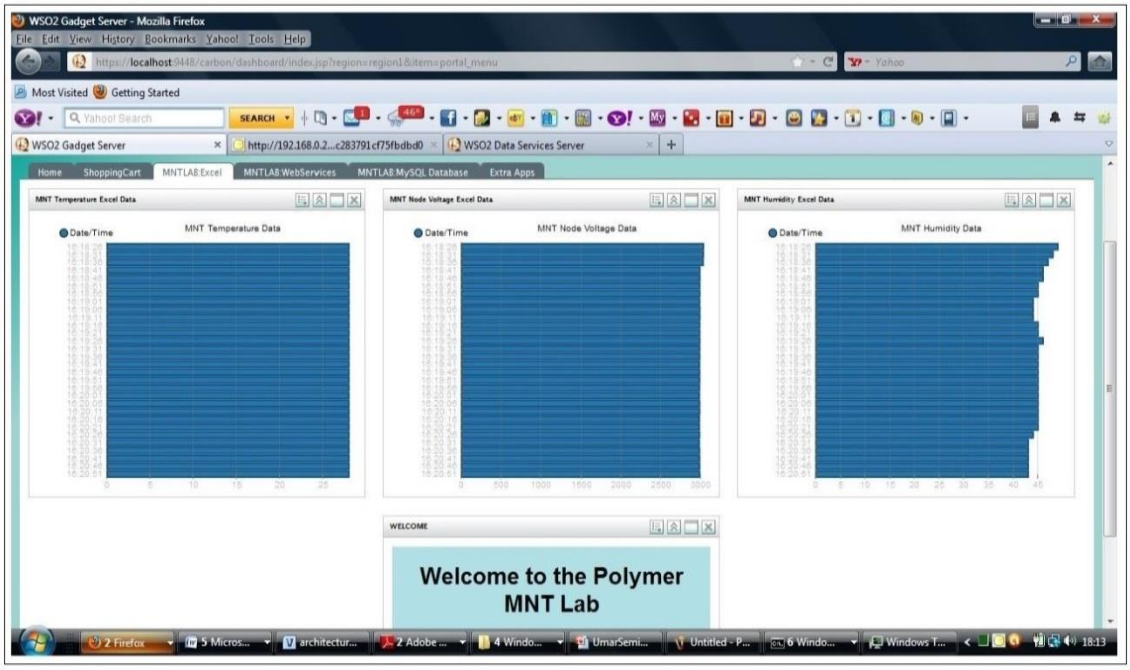


Figure 6.38 Historical data shown in bar chart format using gadgets.

6.7 Conclusion

In this chapter, an experimental test bed system for monitoring the µIM process called the µIMP Monitor System was presented. The proposed system is based on a novel approach using three major technologies: WSN, SOA and GG. The technical issues relating to the system architecture design, component integration, user interfaces and the resulting services have been discussed. The proposed ESB with its graphical user interfaces has proven to be very flexible and easy to configure. It is highly scalable with a potential of hosting a very large number of services at any given time, therefore making it a suitable composition methodology for integrating large number of devices on the shop floor. The GG API has been used with the WS02 gadget sever to develop mini applications for the different stages of the µIM process hence paving the way for future development of complex applications which are modular in nature.

It is worth noting here that the proposed novel approach used in the µIMP Monitor System is not only valid for the experimental test bed scenario described in this chapter, but it can also be used as a generic approach with any other WSN based architecture. The proposed ESB is very flexible and highly scalable as it allows the hosting of a very large number of services using different transport and messaging protocols. This makes it an ideal composition methodology for integrating WSNs and other equipment in various environments from various vendors. The GG API used for the exposing of the different stages of the µIM process on the web can be adapted very easily to expose the functionality of other processes in different application environments. As part of future work, the aim is to monitor the complete µIM process by linking it with other parts of the business enterprise using business process modelling. In the next chapter, a framework for allowing the different applications and their hardware in the µIMP Local Monitor Layer to communicate with each other using Business Processes will be presented.

Chapter 7: µIM Monitor Business Processes Framework

7.1 Introduction

With the recent advances in the computing and communication features of sensors, actuators, and production devices on the factory shop floor, the manufacturing process has evolved to face the challenges posed by increasingly shorter product lifecycles and increased product quality. This has given rise to complex and distributed manufacturing systems consisting of not just only machines and their associated equipment, but other smart devices (WSN nodes, Radio Frequency Identification (RFID) tags, Ethernet and Wireless DAQs) which are equipped with advanced computing and communication capabilities, hence narrowing the digital gap between the computing world and real world products. These advances have been promoted through the Internet of Things (IoT) concept through which the notion of smart factories is becoming a fast reality. The conglomeration of the diverse range of sensing entities in the manufacturing environment brings a new perspective to the way we interact with this environment. New open approaches have been developed and used to interact with the devices in this environment and one of the most common approaches used has been the concept of Service Oriented Architecture (SOA). The SOA when implemented through the use of web services allows interoperability amongst the disparate devices and machines on the shop floor, efficient system integration and the automation of business processes in enterprises or departments (Min-Jeong et al., 2007).

In the field of business and enterprise IT systems the SOA has been an established approach for several years and is used primarily to orchestrate and execute business processes (Loskyll et al., 2012). On the other hand when integrating business processes of an enterprise with the technical manufacturing processes, the SOA has to be adapted to the industrial manufacturing and automation technology. The general approach would be to encapsulate all control functions within a manufacturing system as self-contained services and then use an orchestration methodology cascaded down to technical sub processes level to link and orchestrate these services in a

sequential manner. One of the most commonly used standards in Business IT systems to orchestrate services has been the Web Services Business Process Execution Language (WS-BPEL) abbreviated to BPEL (OASIS, 2007) in the rest of this chapter.

Some of the latest research efforts using BPEL in the manufacturing domains look into dynamic service composition and reconfiguration using semantic web and context aware techniques. These latest service orchestration techniques are focusing on how to react to new process variants and changes in contextual information (e.g., failure of field devices, requirements on the consumption of resources) (Loskyll et al., 2012, Tao et al., 2012).

In this chapter, we build upon our earlier work, which presented an architecture for monitoring a typical plastics industry environment using the SOA and WSN. We extend the ESB based architecture by proposing a new framework for integrating the µIM process with other standard business processes of an enterprise using BPEL. We use BPEL to combine event-driven communication and SOA-based business processes in a µIM environment. Business processes created using BPEL are used to process and analyse the data on a distributed architecture as well as giving the capability to link with other standard business processes. The technical issues relating to the system architecture design; component integration, user interfaces and the resulting services are discussed. An initial test scenario using a LABVIEW based simulated process data, the Wireless Sensor Network based environment monitoring system LVWSM, the WSO2 Carbon Platform's ESB (WSO2 Inc, 2011a), and the Business Process Server (BPS) (WSO2 Inc, 2012e) is tested using this approach.

This chapter is organised as follows: section 7.2 presents the proposed integration framework in which the functionality of each stage in the µIM process is abstracted using relevant sensor technology and web services and the integration with other standard business processes of the enterprise is done using the ESB and BPEL. Section 7.3 presents the extended system and functional architectures built upon the architecture described in section 6.3.2 for the purpose of integrating and linking heterogeneous platforms and

environments using BPEL. Section 7.4 presents the design of the WSO2 BPS based BPEL processes, services, and artefacts. Section 7.5 presents the implementation and testing of the proposed architecture. Finally, section 7.6 presents the conclusion of this chapter.

7.2 Proposed Framework of Integration using ESB and BPEL.

The dynamic nature of the services and devices in the IoT environment can cause a considerable amount of state changes. In a plastics industrial environment the most interchanged messages are based on some kind of event, alarm or notification. An example of this could be an alarm associated with the cavity pressure sensor or the notification of the piston position. These changes are associated with an event source and one or more event consumers that react to the event according to a predetermined setting. The software architecture for this type of event driven communication and integration of systems is known as Event-Driven Architecture (EDA) (Minguez et al., 2011).

In a scenario where the plastics manufacturing environment is part of a large supply chain network that manufactures moulded products with a high degree of customization. The timely dissemination of business-relevant events is of great importance in such networks. For example, the material suppliers would be required to meet strict delivery deadlines. In these situations it would be of great benefit for the enterprise to integrate the business workflows with manufacturing processes in order to meet the required response times and flexibility when reacting to manufacturing events (Minguez et al., 2011). This could be achieved by amalgamating the EDA and SOA through the integration of complex events into business processes. The ESB, which was used as the mediation layer in the µIMP Monitor architecture (previous chapter), can be used for this purpose as it merges both the EDA and SOA paradigms for the purpose of integrating and linking heterogeneous platforms and environments.

In this chapter, we propose a new framework for integrating the µIM process with other standard business processes of an enterprise by augmenting the ESB based µIMP Monitor architecture with business processes using BPEL. Figure 7.1 depicts the proposed framework of integration in which the µIM

process (divided into its constituent stages) is linked to other business process stages such as purchasing and packaging. In this framework, the functionality of each stage in the process is abstracted using a relevant sensor technology and Web Services. For example in the µIM process, the WSN technology and Web services are used to abstract the conditions of the environment whereas the functionality of the µIM machine can be abstracted through the use of a NI High Speed DAQ system such as the Compact-RIO and Web Services. Similarly, the business processes for purchasing and packaging can use wireless sensor technology such as RFID and Web services to keep track of the amount of material used or products manufactured. For each stage the services can be made up of composite services which can be any process parameters like the temperature, pressure, and screw position provided by the sensor technology. The composite services could be made up of further services for example screw position can have the velocity and displacement services. Similarly the standard business processes can be made up of composite services which depict the behaviour and functionality of the underlying process.

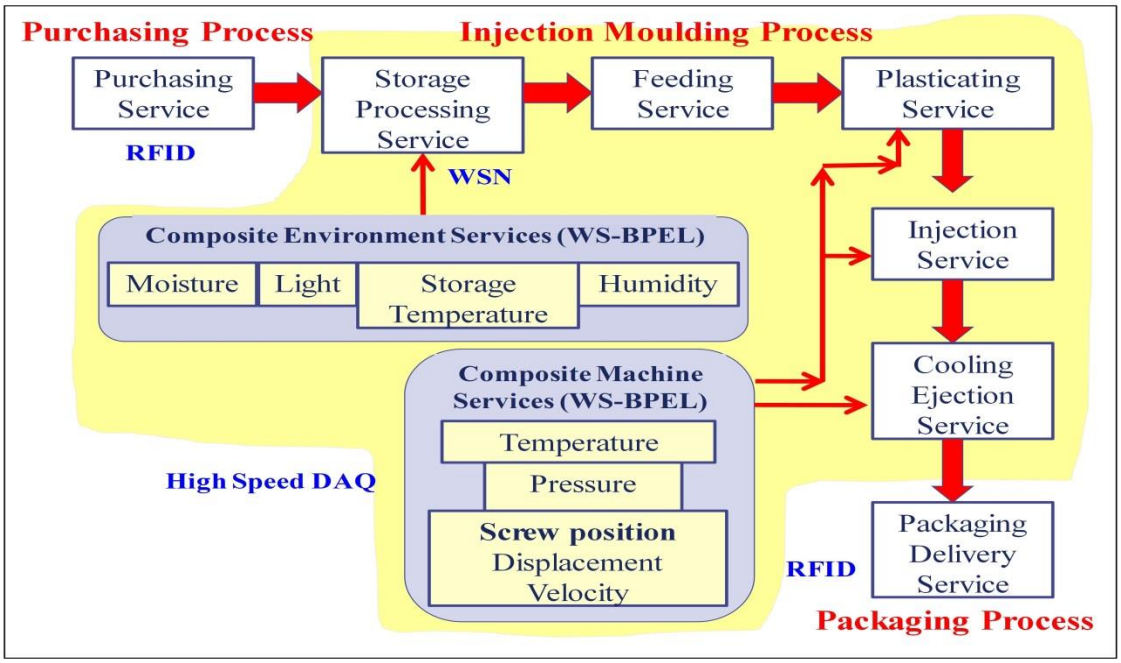


Figure 7.1 Proposed integration framework using sensor technology, Web Services, and BPEL.

Once the services and composite services have been created for each process in each stage, these can be linked with each other using BPEL. Other standard business processes of an enterprise can be integrated in a similar

fashion such as the purchasing, storage, and packaging. BPEL acts as the "glue" to bind these Web Services (depicting the functional behaviour of smaller business processes) in each stage hence allowing the formation of a single unified business process. The composition, binding and orchestration of services in order to fulfil a processes needs allows the formation of a loosely coupled process monitoring system.

7.3 µIMP Monitor Architecture Design using Business Processes

The applications in the µIMP Local Monitor Layer can potentially give rise to a large number of services and composite services driving a large number of other services, business processes and even applications. The resulting plethora of services from the various sensors in the environment and machines can become difficult to manage and orchestrate. Therefore, in order to organise and orchestrate these services into a cohesive manufacturing business process an optimal method of management and orchestration is needed. The µIMP Monitor system architecture presented in the previous chapter uses the ESB as the unified method of data integration between the various layers of the architecture. The ESB acts as the messaging and mediation layer between the frontend web applications and the backend hardware based monitoring applications. In order for the µIMP Monitor system to function as a cohesive entity it needs to have the ability to allow the applications in the µIMP Local Monitor layer to communicate amongst themselves and with the frontend end web applications. For this to materialise the various services in all the layers of the architecture need to be linked and orchestrated in a unified manner. This can be achieved by using BPEL; one of the most commonly used standards in Business IT systems to orchestrate services.

7.3.1 Design of µIMP Monitor System Architecture using Business Processes

In this section, the µIMP Monitor system architecture is presented which has been further extended to include business process orchestration using BPEL. Figure 7.2 shows the extended version of the µIMP Monitor system architecture and it consists of four layers, which are described, in more detail in section 6.2.2. A number of business processes need to be created which can link the services from the different applications in the µIMP Local Monitor layer as well

as the frontend µIMP Web Layer application services. These business processes could be even cascaded down to technical sub processes level to allow the linking and orchestration of the services provided by the individual hardware components such as temperature sensors or machine injection piston in a sequential manner. Once the business processes have been created, the orchestration can then be initiated in the µIMP Global Monitor Layer using customised automated tasks. The WSO2 ESB has a component called “Tasks” which can be used to allow the automation of any piece of software code or services using a timer.

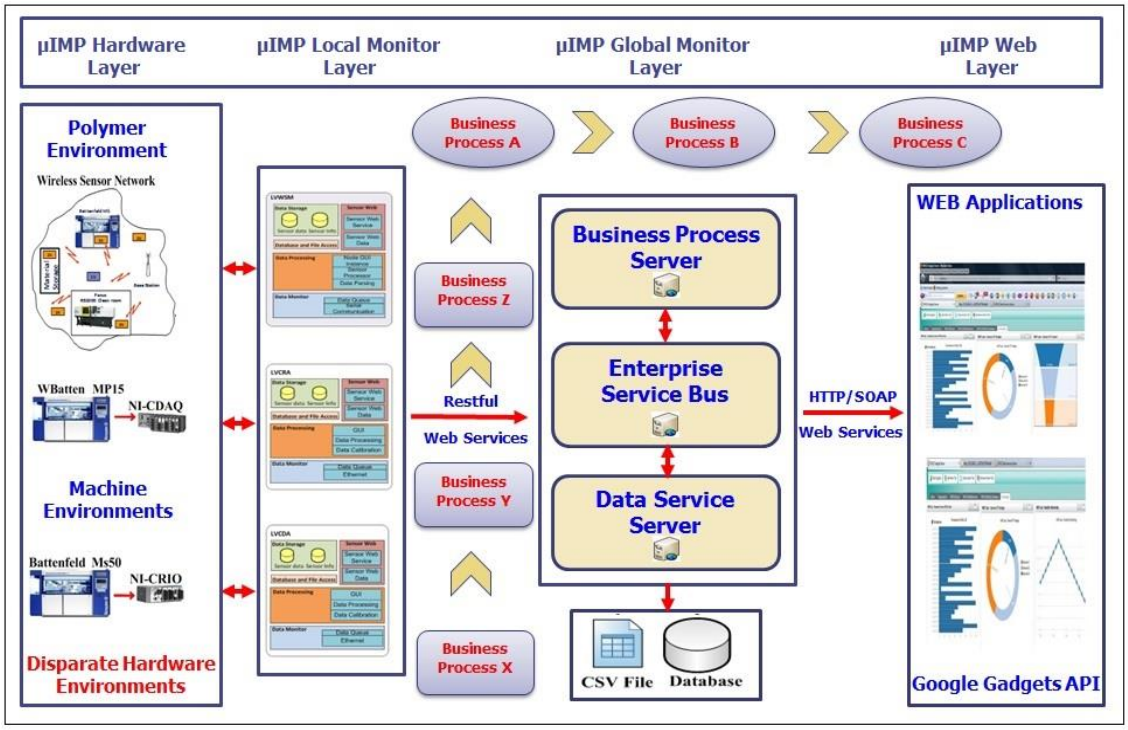


Figure 7.2 µIMP Monitor extended system architecture using business processes..

7.3.2 Design of µIMP Monitor Functional Architecture using Business Processes

Figure 7.3 shows the extended functional architecture design of the µIMP monitor with the WSO2 ESB at its core. The WSO2 ESB has been proposed as the unique method of communication between the different applications in the µIMP Monitor project. The functional architecture consists of four layers, which have already been described, in more detail in section 6.3.2. The µIMP Global Monitor Layer has been modified to allow the use of business processes and these changes are explained in detail in the following section.

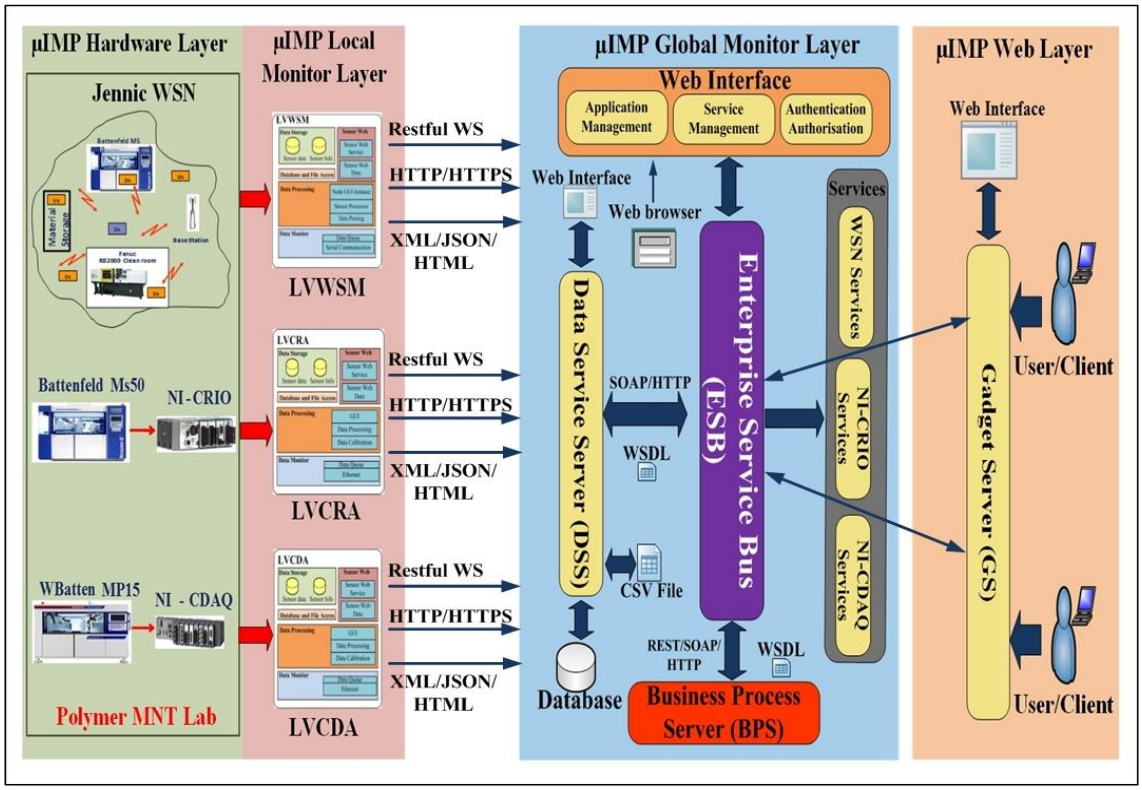


Figure 7.3 µIMP Monitor extended functional architecture using business processes.

7.3.2.1 *µIMP Global Monitor Layer using Business Processes*

The µIMP Global Monitor Layer has been implemented using the WSO2 ESB at its core, which acts as the common data communication, and messaging layer between the different local applications and frontend web applications. The extended version of the µIMP Global Monitor Layer consists of five main components; the Data Service Server (DSS), the ESB, the Web Interface, the Service Registry/Repository, and the Business Process server (BPS). All these components have been implemented using the WSO2 SOA suite of tools.

Data Service Server (DSS): The DSS server described earlier in sections 6.3.2.3 hosts Data Services, which it creates from the data stored in external CSV files as well as MySQL databases. The data services created by the DSS are consumed by the GG web applications to serve as a benchmark for monitoring and detecting trend patterns. The BPS server from time to time will also invoke the data services on the DSS server in order to get the historical data for the various environmental and machine sensors.

Enterprise Service Bus (ESB) Server: The ESB described earlier in sections 6.3.1 allows the communication between service components of a

distributed system via standards-based adapters and interfaces. With the augmentation of the BPS to the µIMP Global Monitor Layer the ESB also gets the ability to initiate business process orchestration. The ESB uses tasks to invoke business processes hosted on the BPS server. A task in the ESB allows the execution of a piece of code or a service triggered by a timer. Tasks can be scheduled in the ESB to execute at variable frequency as well as a defined number of times.

Business Process Server (BPS): The BPS Server is also based on the WSO2 Carbon platform (WSO2 Inc, 2011b) and runs on a dedicated PC machine as part of a distributed system. The BPS is powered by the Apache Orchestration Director Engine (ODE) (Apache, 2012c) BPEL engine. It allows the deployment of business processes developed using the BPEL standard as well as serving as the business process management and hosting environment for the SOA. The orchestration of the business processes is initiated by the ESB, which uses tasks to invoke business processes hosted on the BPS server. Once a business process has been initiated, depending upon the nature of the business process various services can be invoked at regular intervals for getting the latest data from the various sensors in the environment and machines.

7.4 Design of WSO2 based Business Process Services and Artefacts

In order to allow the integration of business processes into the µIMP Global Monitor Layer, the ESB was augmented with the BPS. This allowed business processes to be deployed and orchestrated in this layer. A number of business process services along with BPEL based business process workflows were created using the Eclipse based WSO2 development studio (WSO2 Inc, 2012d). Other artefacts such as the deployment descriptor file (used to configure one or several processes to use specific services); WSDL files for the business process were created to allow the complete business processes to be deployed on the BPS server. A number of proxy services were created on the ESB, which were invoked by the business processes on the BPS server. Tasks were also created in the ESB to allow the automated invocation of business processes deployed on the BPS server.

7.4.1 Design of WSO2 based Business Processes

To allow the deployment and orchestration of business processes on the BPS server the WSO2 developer studio was used to develop a number of business processes. A business process can be viewed as a collection of related activities or tasks that model a business use case and produce a specific composite service, which is a composition of several Web services. BPEL was used to implement the business processes in the µIMP Global Monitor Layer. BPEL can be used to describe business processes in two different ways either executable or abstract business processes.

Executable business processes: Are used to develop business processes with the exact details and can be executed by a BPEL process server such as the WSO2 BPS.

Abstract business processes: Are used to process templates or public message exchanges without including the specific details of process flows. Abstract business processes are not executable and are rarely used.

In this project, executable business processes were used with each business process having the following setup:

- An associated BPEL based business process workflow
- A WSDL file for the business process
- WSDL files for other services to be invoked by this process
- A deployment descriptor file.

Once a business process is completed and deployed on the BPS server, it is seen as a service by other users and processes and can be interacted with like a normal web service.

7.4.2 BPEL based Business Process Layout

A typical BPEL process consists of a number of steps and each step is called an activity. BPEL supports simple and complex activities.

Simple Activities: A simple activity can represent basic constructs, which are used for common tasks such as:

- Invoking services

- Waiting for client response and replying to clients
- Manipulating data variables
- Indicating faults and exceptions
- Waiting for a specified period of time
- Terminating the process

Complex Activities: BPEL supports several structured activities, which can be used to combine simple activities to create complex business processes. The most important of these structured activities are:

- Sequence activity for defining a set of activities that will be invoked in an ordered sequence
- Flow activity for defining a set of activities that will be invoked in parallel
- The Conditional (<if>) construct for implementing branches
- The While, repeat, and for each loops
- The Pick construct for select one of a number of alternative paths

7.4.2.1 *BPEL process structure and external service invocation*

A typical business process created using BPEL begins with the process definition written in XML using the <process>root element as shown in Figure 7.4. The <process> element has a top-level main <sequence> element. Within the main sequence, the process first waits for the incoming message to start the process. This wait is implemented using the <receive>construct. Figure 7.5 shows the code excerpt for both the main sequence and the receive construct. The process then invokes the related services, using the <invoke>construct. Invocations can be done sequentially or in parallel. If the services are required to be invoked in a sequence then we simply use an <invoke>construct for each invocation and the services will be invoked in that order. If the services need to be invoked concurrently then the <flow> construct can be used to invoke them in parallel. If a service needs to be invoked conditionally then we can use the <If> construct to conditionally invoke or execute a certain branch of the code.

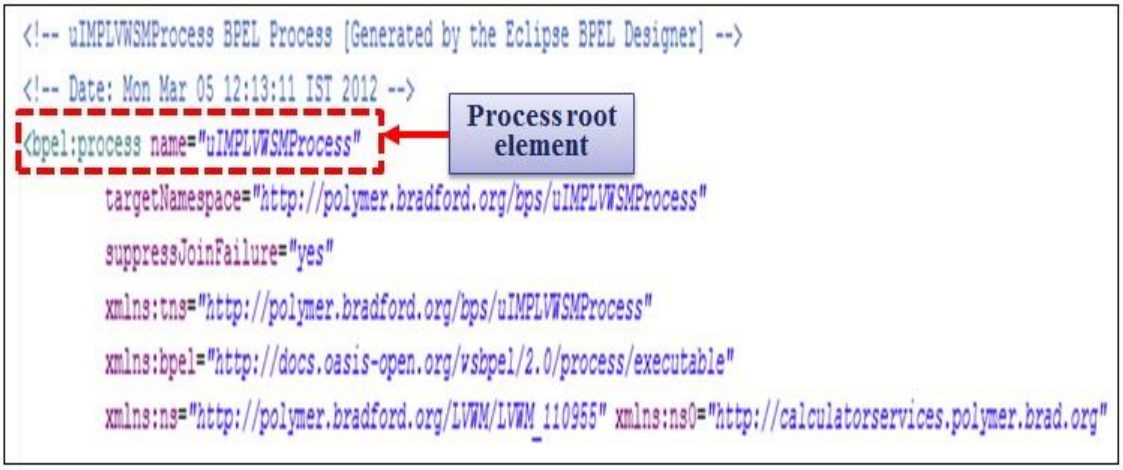


Figure 7.4 Start of a BPEL process file.

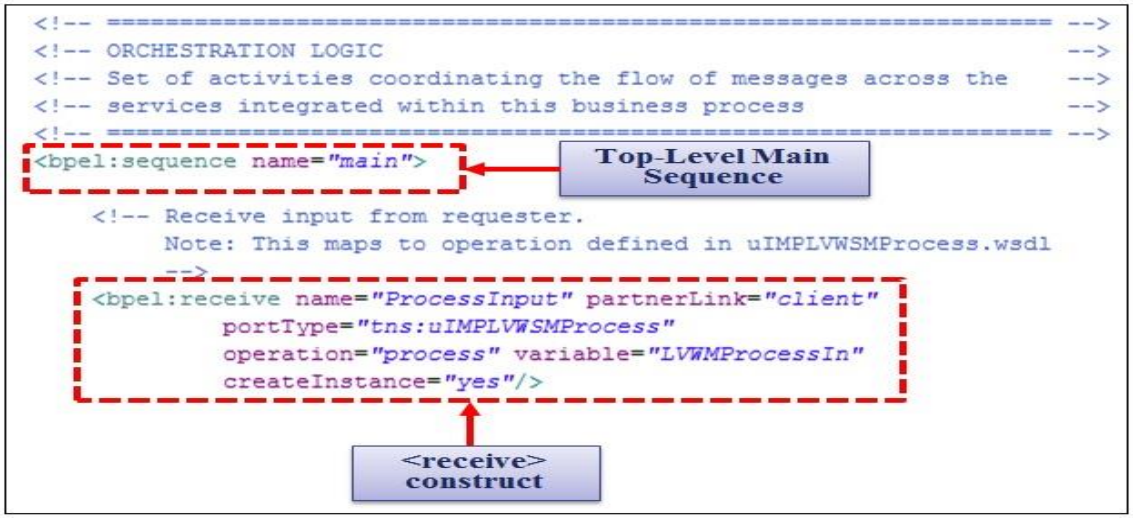


Figure 7.5 BPEL process main sequence and receive construct.

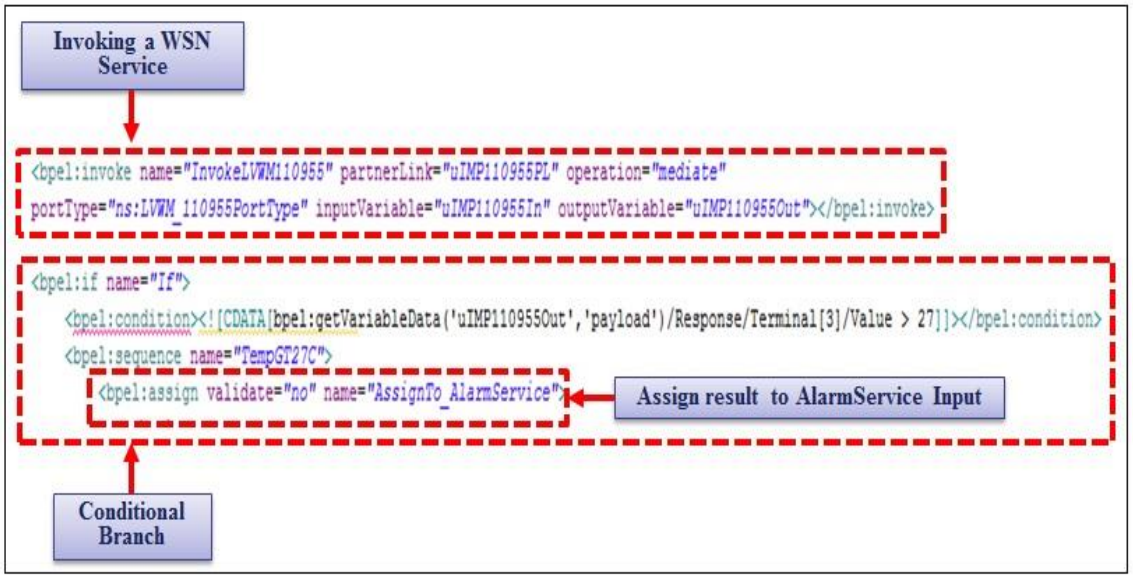


Figure 7.6 Invoking a WSN service to retrieve sensor data.

Figure 7.6 shows a code excerpt for invoking a service, which retrieves temperature data from a WSN node (address: 110955). The conditional <If> construct is used to verify if the temperature is above a certain value before assigning the value to an alarm service.Other than invoking services the BPEL process can also receive invocations from clients. One of the clients will be the user of the current BPEL process, who makes the initial invocation. Other clients can be asynchronous services that are invoked by the BPEL process and which make a callback to return the data from the invoked service.

7.4.2.2 *BPEL Process Interaction with external services*

As mentioned in the previous section BPEL processes interact with external services by either invoking operations on external services or by receiving invocations from external clients. The links used by BPEL to interact with all services and clients are called partner links.

Partner Links: The code excerpt in Figure 7.7 shows an example of defined partner links. Partner links can be links to clients, which can invoke the BPEL process and are sometimes called client partner links. Each BPEL process has to have at least one client partner link, because there has to be a requestor client that first invokes the BPEL process. The other partner links are links to asynchronous services that are invoked by the BPEL process and are

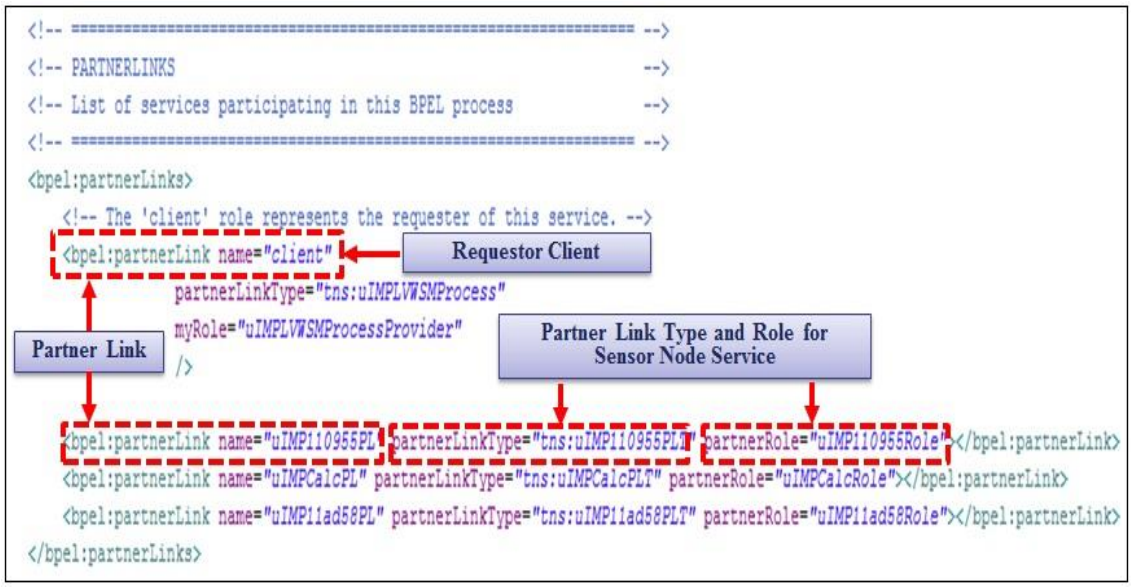


Figure 7.7 BPEL process clients and partner links, types and roles.

sometimes called invoked partner links. A typical BPEL process will have at least one invoked partner link because it will most likely invoke at least one service. The invoked partner links can become client partner links when the process invokes an operation on an asynchronous service. The asynchronous service will later on invoke a callback operation on the process to return the requested data. In this project, we have used synchronous services, which return any calculated results back to the requestor using the <reply> construct.

Partner Link Type and Role: A BPEL process invokes the operations on a service through the services portType. The BPEL process also has to provide a portType through which the service invokes the callback operation. Figure 7.8 shows the portTypes for the uIMPLVWSM process and the sensor node Web Service. From the viewpoint of the uIMPLVWSMprocess it requires “Sensor 110955 portType” on the sensor node Web Service and provides “uIMPLVWSMportType” to the sensor node Web Service. From the perspective of the sensor node Web Service it offers “Sensor 110955 portType” to the uIMPLVWSMprocess and requires “uIMPLVWSMportType”.

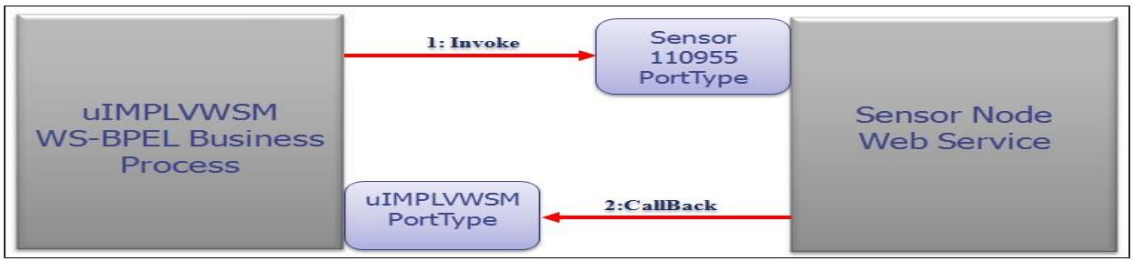


Figure 7.8 PortTypes for the BPEL process and sensor node web service.

In situations when a service is invoked by the business process and vice versa, there is a need to select a certain perspective. In our example, we can select either the uIMPLVWSM process perspective or the sensor node Web Service perspective. In both the cases, we need to describe the portTypes required and offered. BPEL uses the partner link type to overcome this limitation by allowing to model relationships between the various parties. Instead of taking a certain perspective, BPEL allows the defining of roles. A partner link type must have at least one role and have at the most two roles. For each role, a portType needs to be specified that is used for interacting with the service or the process. In short, a partner link type declares how two parties interact with each other and what they have to offer in terms of their role.

7.4.2.3 *BPEL Process Variables*

In a BPEL business, process messages are exchanged between the process, client, and various services. This happens when an operation is invoked on a service or vice versa. In either case, a result is returned which needs to be stored for various reasons such as subsequent invocations, use the result as is, or extract certain data. BPEL variables can also be used to hold data that relates to the state of the process. Variables can store WSDL messages, XML schema elements, or XML schema simple types. The <variables> element is used to group all the variable declarations in one place as shown in Figure 7.9. BPEL variables are used in the <invoke>, <receive>, and <reply>constructs to specify the input and output messages for invoking operations on partner services. To copy data between variables, expressions, and partner link endpoint references, BPEL provides the <assign> activity. The assign activity uses the <copy> command to copy data from a source to a destination. Figure 7.6 shows the assign activity used to copy the result from the output variable of the LVWM110955 sensor node service to the input variable of the AlarmService.

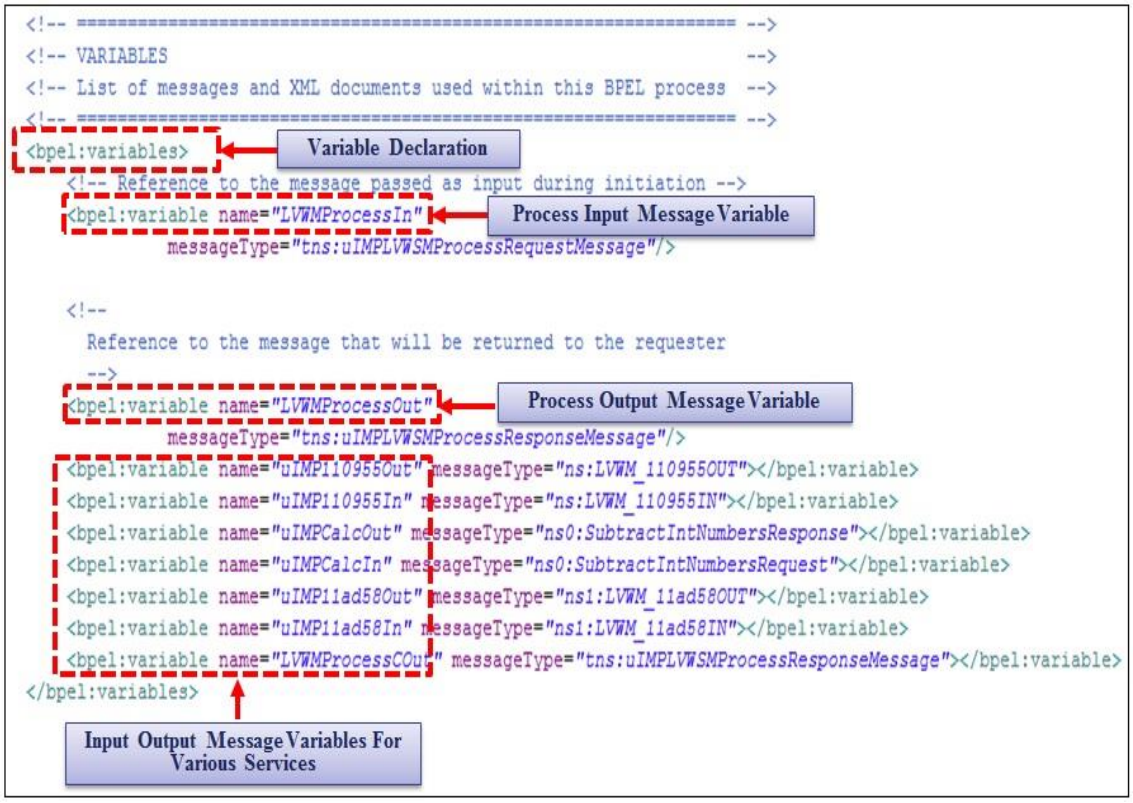


Figure 7.9 BPEL process variable declaration.

7.4.3 Business Process WSDL file

BPEL business processes are seen as services by external services and clients. Therefore, each BPEL process needs a WSDL document. As mentioned before a client will usually invoke an operation on the BPEL process to initiate the process. The BPEL process WSDL file specifies the interface for this operation needed by the client. The WSDL file also specifies all message types, operations, and port types a BPEL process offers to other partners. For each BPEL process created in this project a WSDL file was also created which defines all the above attributes. The complete WSDL files for the BPEL processes and partner services can be found in Appendix F.

7.4.4 WSDL files for services to be invoked

As mentioned before a BPEL process will invoke services either sequentially or in parallel. Each service that is invoked and used in the BPEL process has to have a WSDL document. The BPEL process will invoke an operation on the services used in the process. The service WSDL file specifies the interface for the various operations for each service in the process. The WSDL file also specifies all message types and port types a service offers to the BPEL process and other services. A number of proxy services were created to get data from the various RESTful web services provided by the LVWSM application, which in turn extracted sensor data from sensor nodes. These proxy services were deployed on the ESB server. For each proxy service, a WSDL file was also created. The complete WSDL files for the BPEL processes and the services can be found in Appendix F.

7.4.5 Deployment Descriptor File

The WSO2 BPS and Apache ODE use a deployment descriptor file named deploy.xml to configure one or several business processes to use specific services. For each BPEL business, process a descriptor file was created. The deployment descriptor file is used during the deployment phase of the business process in which the process engine loads all documents from this file. By loading the documents, the business process can reference processes, service and schema definitions using fully qualified names, and import based on namespaces instead of locations. The deploy.xml also has the binding information for partner links to concrete WSDL services. Without the descriptor

file the BPEL engine cannot determine the exact details of the partner services. In the deploy.xml, the <service/> element defines the exact details of all partner services that interact with the BPEL process. A deployment descriptor file was created for each BPEL process and can be found in Appendix F.3.

7.5 Implementation and Testing

This section focuses on the implementation and testing of the proposed framework for integrating the μIM process with other standard business processes of an enterprise using BPEL. The extended ESB based μIMP Monitor architecture has been implemented and tested in the centre for Polymer MNT at the University of Bradford. An initial framework has been implemented in the μIMP Global Layer for the application of BPEL to allow the orchestration of services and to link with other business processes in the μIM environment. A simple test scenario was implemented in which a BPEL business process was created and used to first invoke a service to initiate a WSN based process, which monitored the environmental conditions (temperature and humidity) in a material storage cupboard. Based on the sensor readings in the material storage cupboard, the BPEL business process then invoked two services to initiate two different processes. A material drying process was initiated if the humidity in the material storage cupboard was above a certain threshold level. Similarly, an alarm process was triggered if the temperature in the material storage cupboard was above a certain threshold level.

7.5.1 Implementing the Environmental Monitoring Process

The LVWSM process monitoring system presented in chapter five was used to monitor the humidity and temperature in a materials storage cupboard. It used a Jennic JN5148 wireless sensor node to acquire the temperature and humidity readings in the material storage cupboard. The JN5148 sensor nodes have an integrated Sensirion SHT11 single-chip multi-sensor module for the measurement of temperature and relative humidity. The two sensor readings were extracted and converted into a RESTful web service.

7.5.1.1 *Environmental Monitoring Process Proxy Services*

A number of proxy services were created as described in section 6.5.2 for the different sensor nodes in the Jennic WSN. These proxy services were

implemented on the WSO2 ESB server in the μIMP Global Monitor Layer and exposed through endpoints. When these proxy services are invoked, the BPEL business process sends a request to the relevant RESTful web services to retrieve the sensor readings. Figure 7.10 shows a code excerpt for a proxy service created to retrieve data from the sensor node with address 110955.

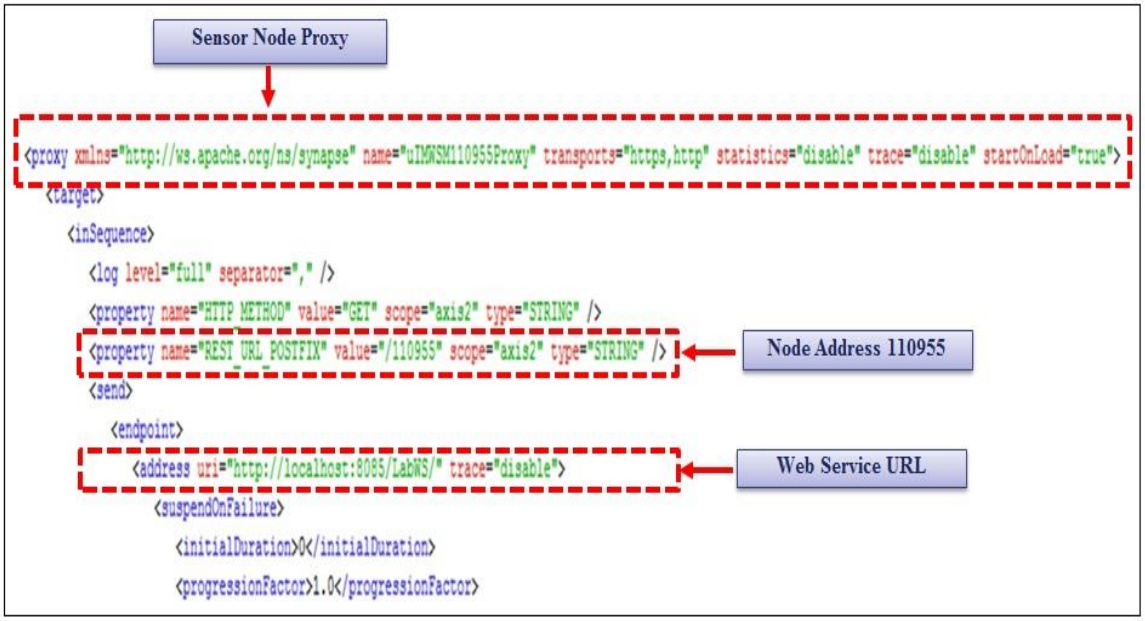


Figure 7.10 Sensor node proxy service.

7.5.1.2 *Environmental Monitoring Process WSDL files*

Each proxy service running on the WSO2 ESB server has an associated WSDL file. A WSDL file provides a description of how the proxy service is to be called, the parameters it expects, and the data structures it will return. It exposes the service to the outside world as a collection of reusable network ports and endpoints. A port is defined by linking an IP address with the reusable port bindings. The data is exchanged using abstract descriptions and the port types give the abstract collections of supported operations. BPEL processes and clients that connect to this proxy service can determine what operations are available by reading the WSDL file. The BPEL process and clients using SOAP/XML and HTTP can call these operations. The XML code excerpt in Figure 7.11 shows the WSDL file created for the “LVWM_110955” proxy service used for retrieving sensor data from the “LabWS” RESTful web service. In this excerpt, the request and response elements as well as the portType used to interact with the proxy service are defined. Figure 7.12 shows a port binding over HTTP, which is one of the three reusable port bindings implemented. It

specifies the HTTP method used and operation, which can be invoked on this service. The code excerpt in Figure 7.13

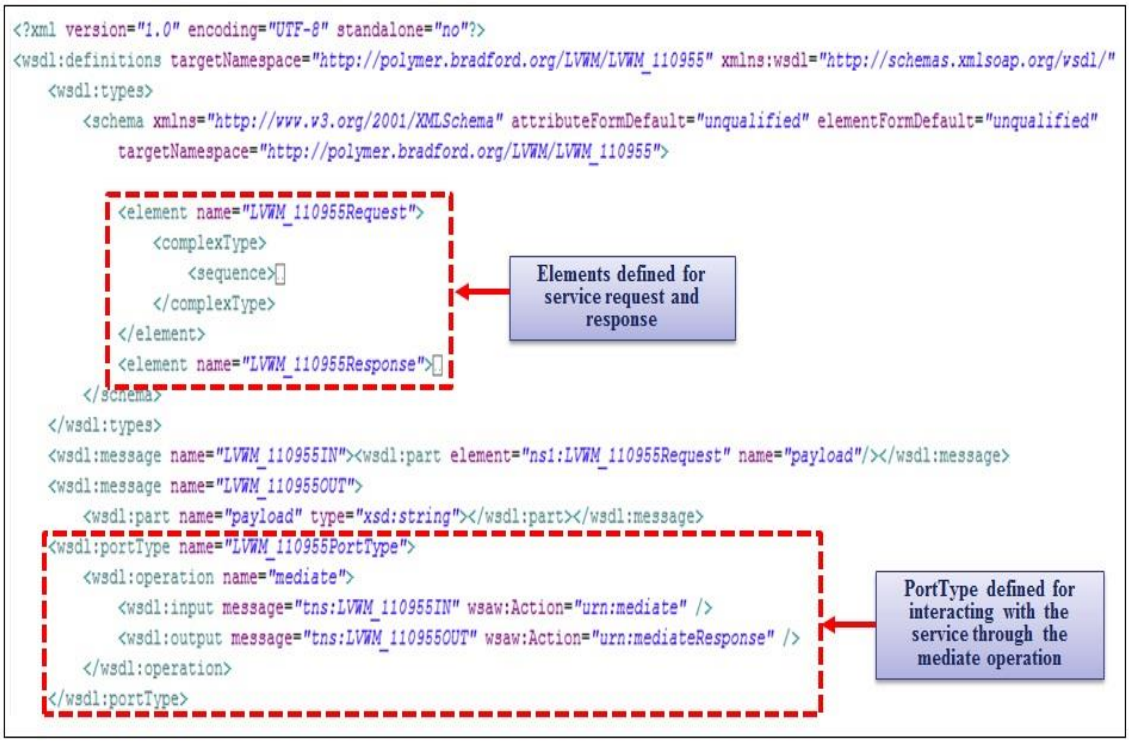


Figure 7.11 Sensor node 110955 WSDL file.

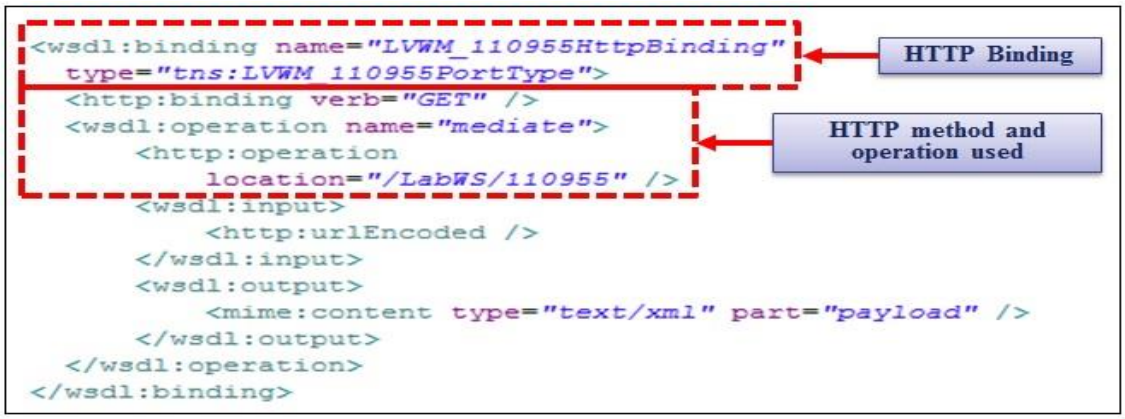


Figure 7.12 Port binding for the sensor node 110955.

shows the service name as well as the defined ports which link the IP addresses to each of the three port bindings. The graphical view of the complete WSDL file for this service is detailed in Figure 7.14. The complete XML WSDL file for this service as well as for the other sensor nodes can be found in Appendix F.

Figure 7.13 Sensor node 110955 service ports.

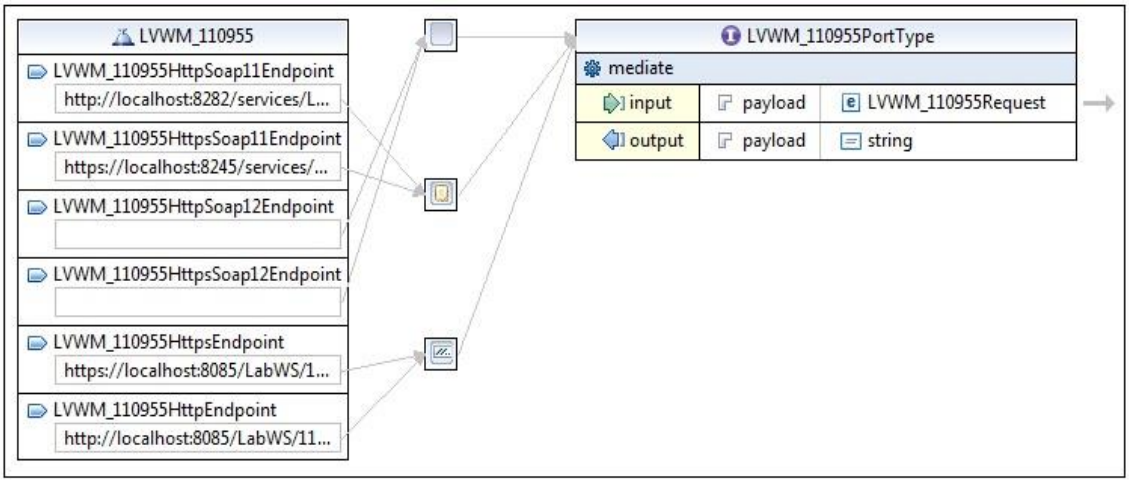


Figure 7.14 Graphical view of the sensor node 110955 WSDL file.

7.5.2 Implementing the Material Dryer Process

The material drying process was simulated by a simple LABVIEW based simulator application called “MaterialDryerSim” as shown in Figure 7.15. This application takes the humidity value as an input and compares this with a variable humidity threshold value. If the received humidity value is greater than the set threshold value, then the material drying process is started. The drying process is simulated by flashing the green LED to indicate that the material drying process is currently running. The humidity value decreases every 5 seconds until it goes below the threshold value. At this point the material drying process stops which is indicated by the red LED and the HUMIDTY IN LIMITS green LED turns ON. The humidity value will stay below the threshold until it is

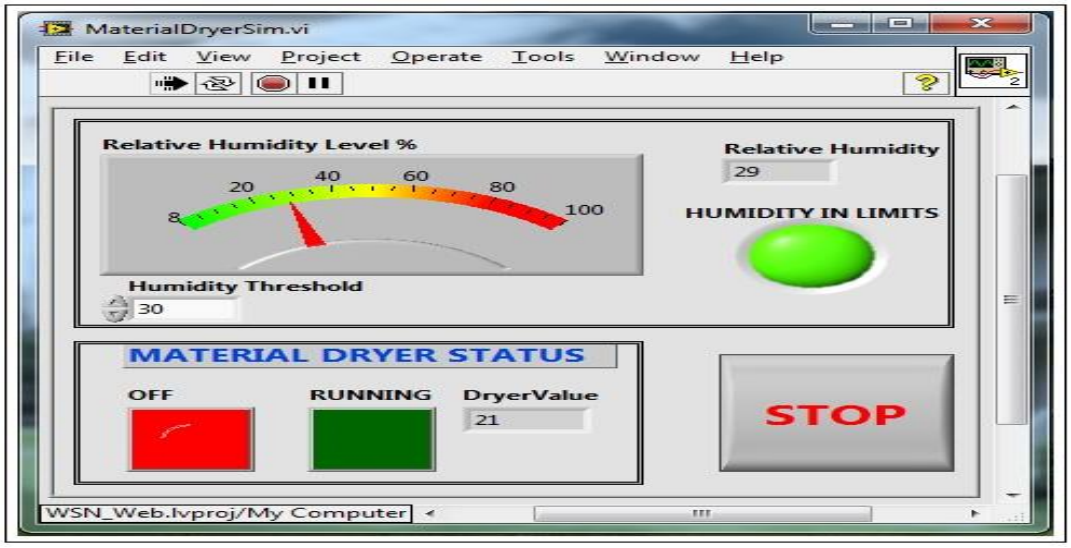


Figure 7.15 Material dryer process simulator.

stopped or the threshold is decreased. If the threshold is decreased it will start and run the drying process until the relative humidity is within the threshold limits. The DryerValue in the application is incremented every 1 second to indicate that a separate drying process is running within the application. It only increments when the material drying process is running. The value indicates the length of time the dryer simulation has been running. The complete LABVIEW code for this application can be found in Appendix D.6.

7.5.2.1 *MaterialDryerSim Restful Web Service*

The MaterialDryerSim application creates a RESTful web service called "uIMBPELDryerWS" for the relative humidity value and runs this web service on

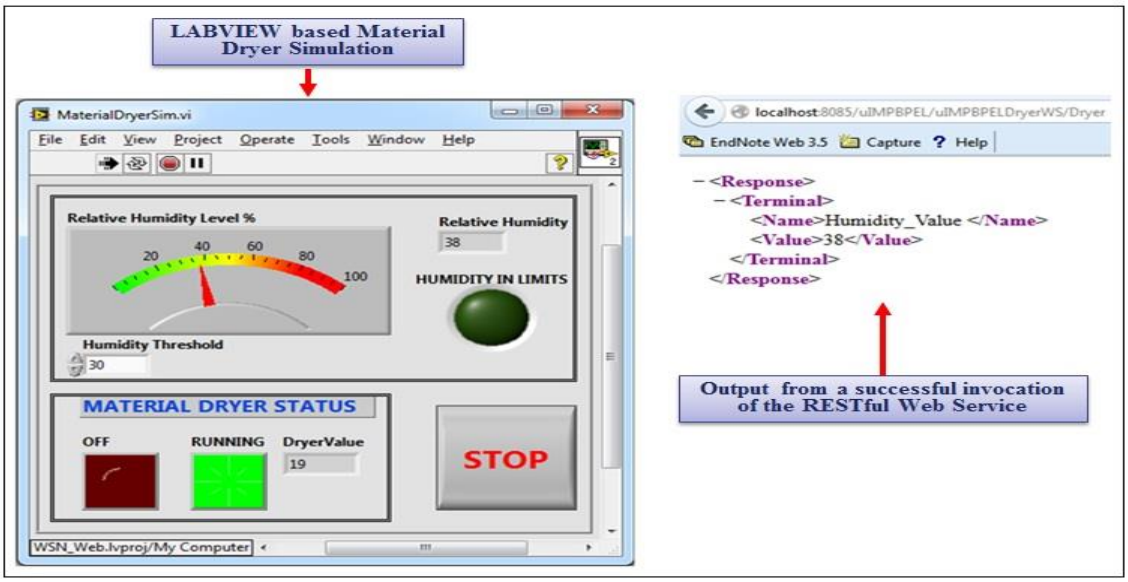


Figure 7.16 Material dryer RESTful web service invocation.

the local LabVIEW Application web server on port 8085. The web service can be invoked through the following URL. http://localhost:8085/uIMPBPEL/uIMPBPELDryerWS/Dryer. Figure 7.16 shows the MaterialDryerSim application running and the output from a successful invocation using the above URL.

7.5.2.2 *Material Dryer Process Proxy Service*

A proxy services were created as described in section 6.5.2 for the Material Dryer Process, was implemented on the WSO2 server in the µIMP Global Monitor Layer, and exposed using an endpoint. When the proxy service is invoked, the BPEL business process sends a request to the relevant RESTful web service to retrieve the Humidity_Value reading. Figure 7.17 shows a code excerpt for the created proxy service.

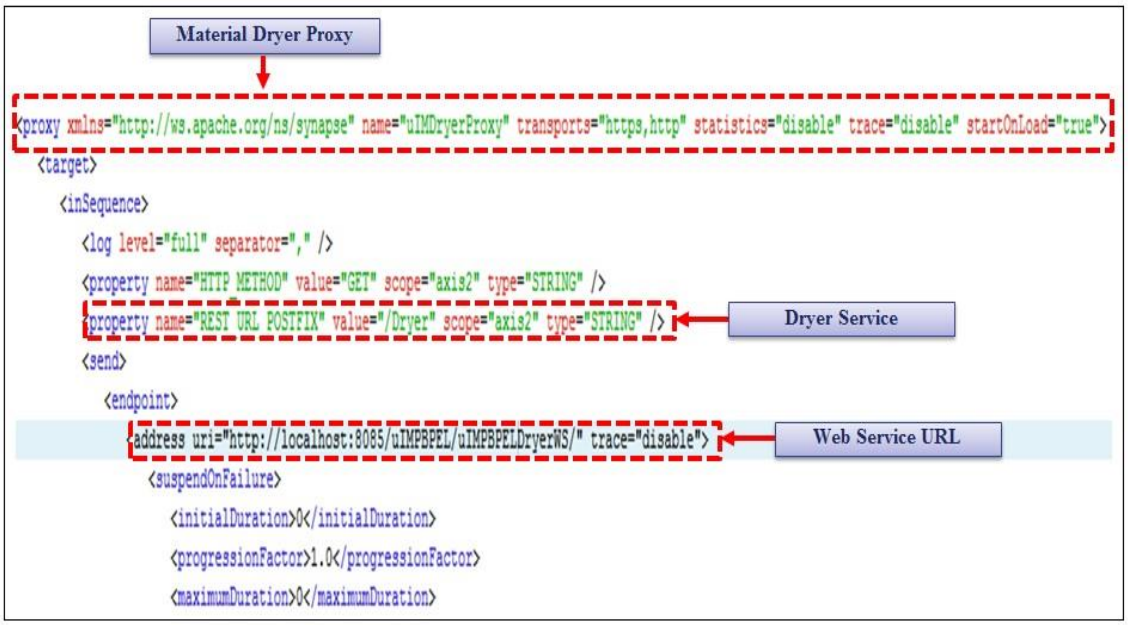


Figure 7.17 Material dryer process proxy service.

7.5.2.3 *Material Dryer Process WSDL file*

An associated WSDL file was created for the Material Dryer proxy service running on the WSO2 ESB server. The WSDL file provides a description of how the proxy service is to be called, the parameters it expects, and the data structures it will return. It exposes the service to the outside world as a collection of reusable network ports and endpoints. Figure 7.18 details a graphical version of the complete WSDL file developed using the Eclipse based WSO2 Development Studio. The complete XML version of this WSDL file can be found in Appendix F.4.2.

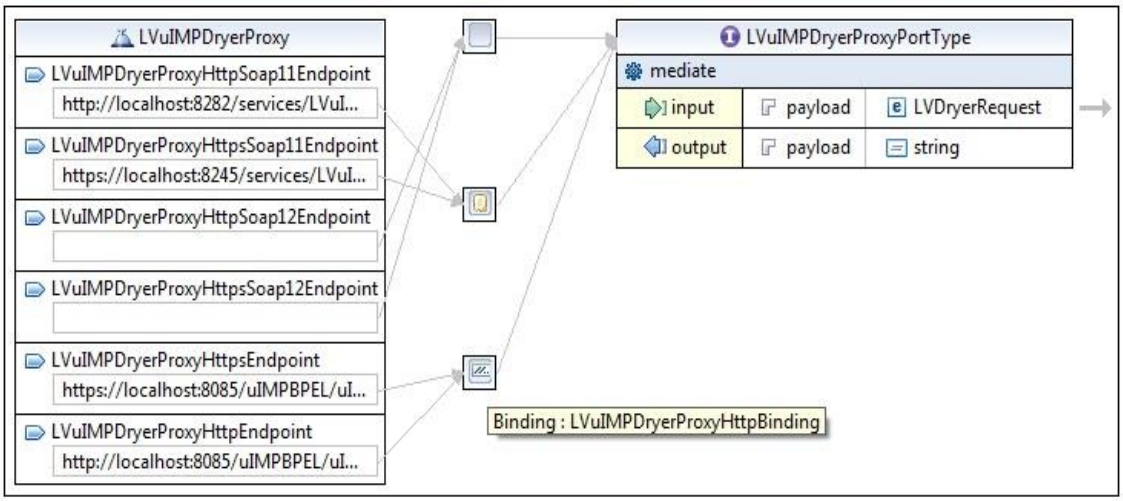


Figure 7.18 Graphical view of the material dryer process WSDL file.

7.5.3 Implementing the Temperature Alarm Process

The temperature alarm process was simulated by a simple LABVIEW based simulator application called “TemperatureAlarm” as shown in Figure 7.19. This application is similar to the material drying process in section 7.5.2, takes the temperature value as an input, and compares this with a variable temperature threshold value. If the received temperature value is greater than the set threshold value, then the temperature alarm process is triggered. The alarm process is simulated by a flashing RED LED which indicates that the alarm has been triggered. The temperature value decreases every 5 seconds until it goes below the threshold value. At this point, the temperature alarm process stops, the GREEN LED turns on, and this triggers the TEMPERATURE IN LIMITS GREEN LED to turn ON. The temperature value will stay below the threshold value until it is stopped or the threshold is decreased. If the threshold is decreased, it will start and run the alarm process until the temperature is within the threshold limits. The AlarmValue in the application is incremented every 1 second to indicate that a separate alarm process is running within the application. It only increments when the temperature alarm process is running. The value indicates the length of time the alarm simulation has been running. The complete LABVIEW code for this application can be found in Appendix D.5.

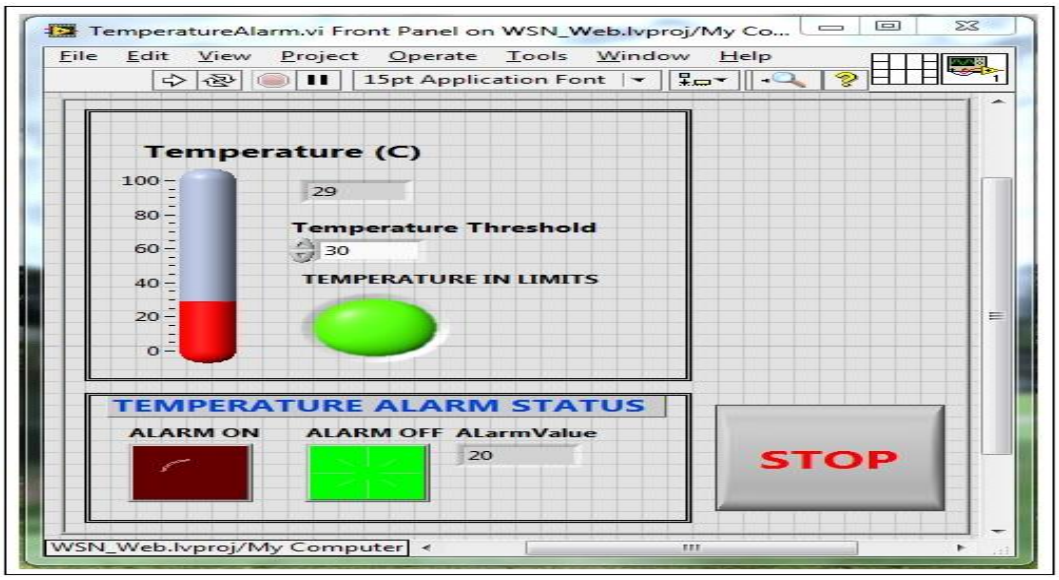


Figure 7.19 Temperature alarm process simulator.

7.5.3.1 *TemperatureAlarm Restful Web Service*

The TemperatureAlarm application creates a RESTful web service called "uIMBPELTAlarmWS" for the temperature value and runs this web service on

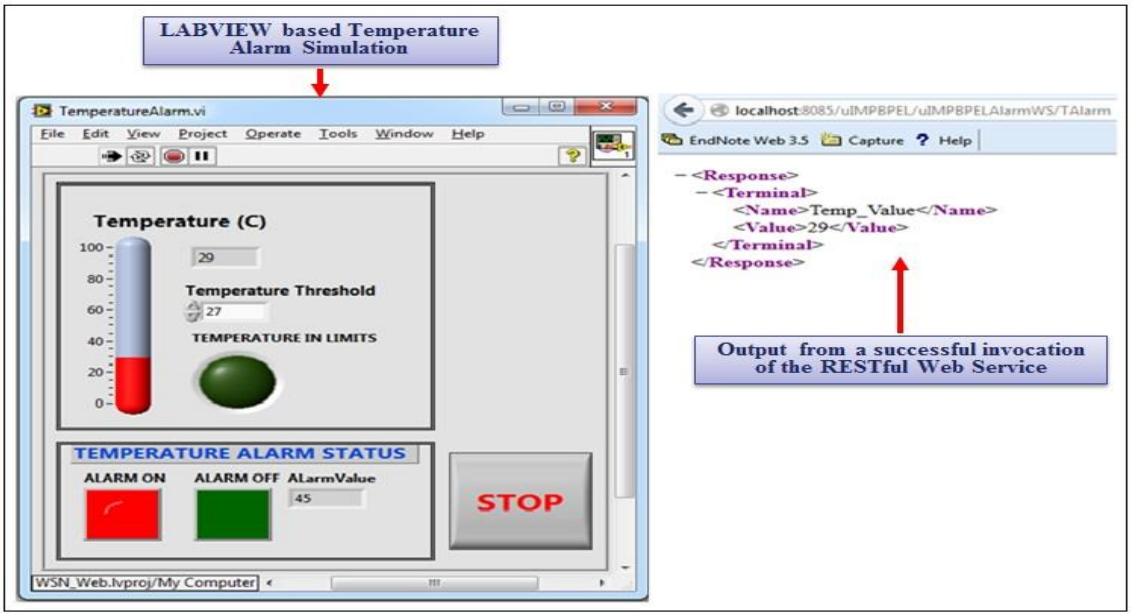


Figure 7.20 Temperature alarm RESTful web service invocation.

the local LabVIEW Application web server on port 8085. The web service can be invoked through the following URL. http://localhost:8085/uIMPBPEL/uIMPBPELTAlarmWS/TAlarm. Figure 7.20 shows the TemperatureAlarm application running and the output from a successful invocation using the above URL.

7.5.3.2 *Temperature Alarm Process Proxy Service*

A proxy services were created as described in section 6.5.2 for the Temperature Alarm Process, was implemented on the WSO2 server in the µIMP Global Monitor Layer, and exposed using an endpoint. When the proxy service is invoked, the BPEL business process sends a request to the relevant RESTful web service to retrieve the Temp_Value reading. Figure 7.21 shows a code excerpt for the created proxy service.

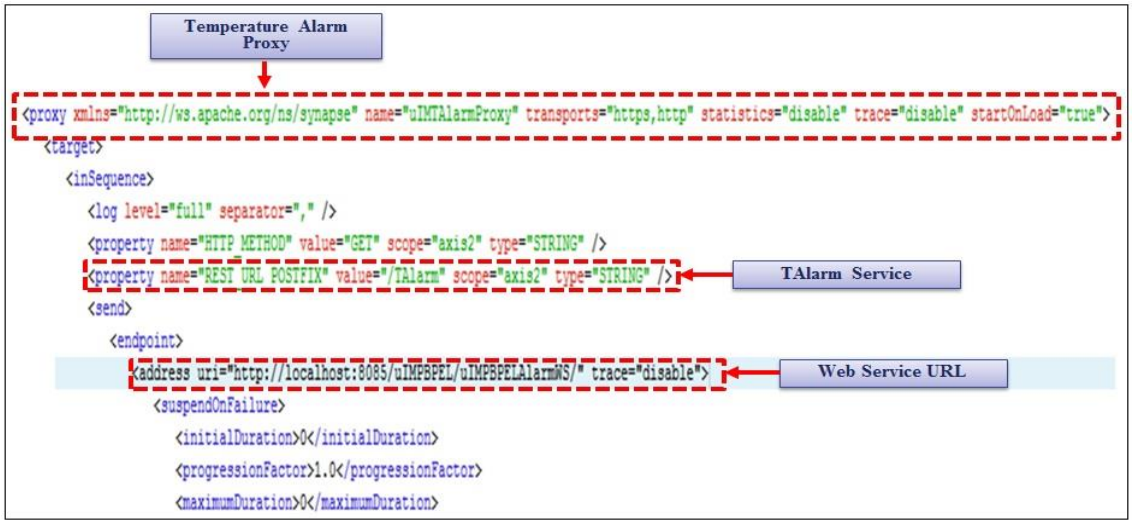


Figure 7.21 Temperature alarm process proxy service.

7.5.3.3 *Temperature Alarm Process WSDL file*

An associated WSDL file was created for the Temperature Alarm proxy service running on the WSO2 ESB server. The WSDL file provides a description of how the proxy service is to be called, the parameters it expects, and the data structures it will return Figure 7.22 details a graphical version of the complete WSDL file developed using the Eclipse based WSO2 Development Studio. The complete XML version of this WSDL file can be found in Appendix F.4.3.

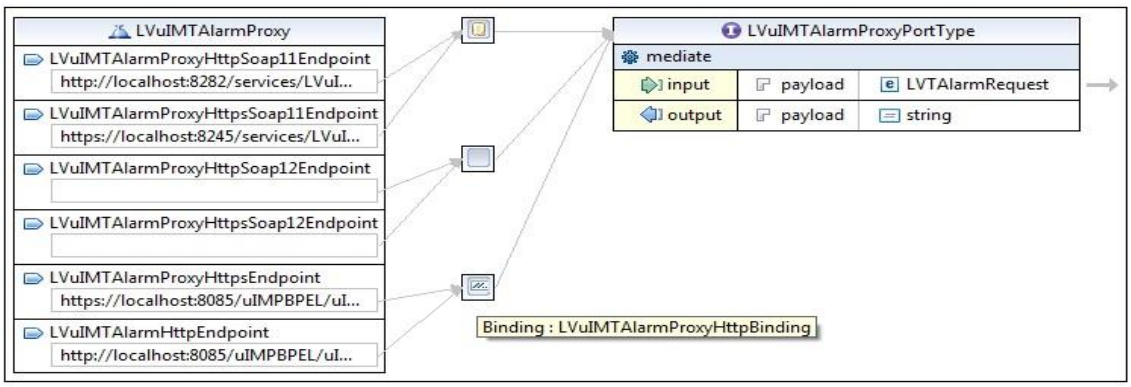


Figure 7.22 Graphical view of the temperature alarm process WSDL file.

7.5.4 Implementing the BPEL Process

The BPEL process design described in section 7.4.2 was implemented using the Eclipse based WSO2 Developer Studio. Figure 7.23 shows the developed workflow called “uIMPLVWSMProcess” to test the initial framework. In this BPEL workflow, a simple test scenario is implemented in which the BPEL business process first invokes the LVWM110955 proxy service described in section 7.5.1. The Jennic WSN based process monitors the environmental conditions (temperature and humidity) in the material storage cupboard. Based on the sensor readings in the material storage cupboard, the BPEL business

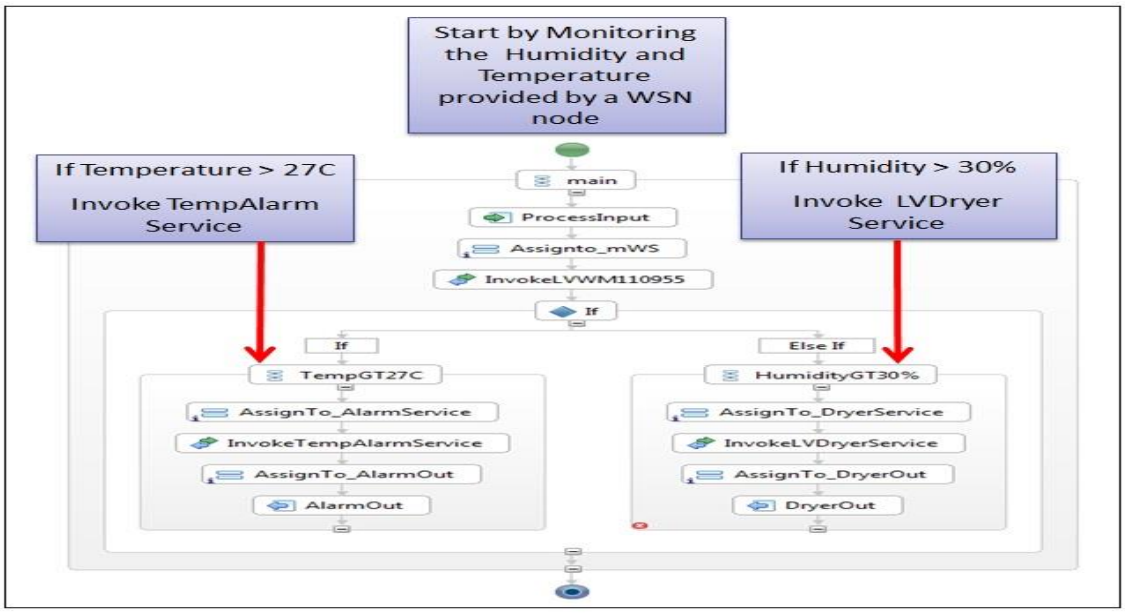


Figure 7.23 uIMPLVWSMProcess BPEL process visual workflow..

process then invokes two services to initiate two different processes. The material drying process described in section 7.5.2 is invoked if the humidity in the material storage cupboard is above 30%. Similarly, the temperature alarm process described in section 7.5.3 is invoked if the temperature in the material storage cupboard is above 27 degree centigrade.

7.5.4.1 *The uIMPLVWSMProcess BPEL Process WSDL file*

External clients and services see every BPEL business process as just another web service. An associated WSDL file was created for this process. The BPEL process WSDL file contains links to all the WSDL files of services to be invoked by the process. Figure 7.24 shows the graphical version of the WSDL file for the developed BPEL process. The complete XML version can be found in Appendix F.4.1.

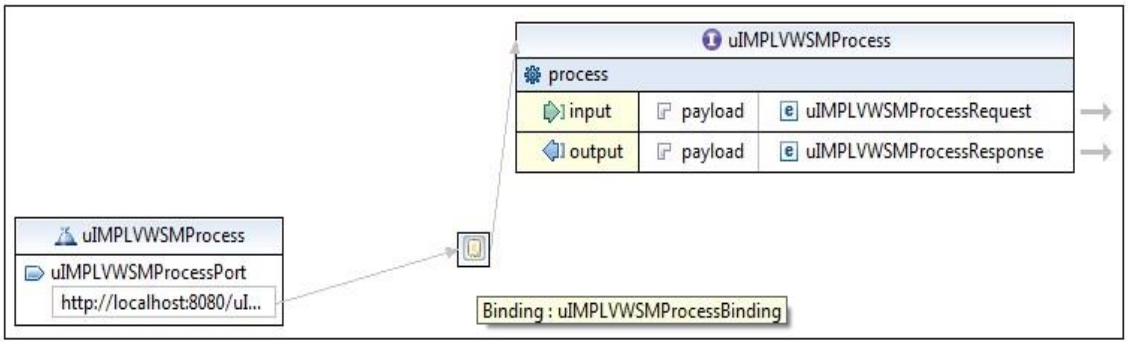


Figure 7.24 Graphical view of the BPEL process WSDL file.

7.5.4.2 *The uIMPLVWSMProcess BPEL Process Descriptor file*

As described in section 7.4.5 the WSO2 BPS and Apache ODE use a deployment descriptor file named “deploy.xml” to configure one or several business processes to use specific services. The deployment descriptor file is used during the deployment phase of the business process in which the process engine loads all documents from this file. The deployment descriptor file created for the uIMPLVWSMProcess BPEL process is detailed in Figure 7.25.

```
<?xml version="1.0" encoding="UTF-8"?>
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
        xmlns:LVDryer="http://polymer.bradford.org/LVSIM/LVDryer"
        xmlns:LVTAlarm="http://polymer.bradford.org/LVSIM/LVTAlarm"
        xmlns:LVWM_110955="http://polymer.bradford.org/LVWM/LVWM_110955"
        xmlns:sample="http://polymer.bradford.org/bps/uIMPLVWSMProcess">
  <process name="sample:uIMPLVWSMProcess">
    <active>true</active>
    <retired>false</retired>
    <process-events generate="all"/>
    <provide partnerLink="client">
      <service name="sample:uIMPLVWSMProcess" port="uIMPLVWSMProcessPort"/>
    </provide>
    <invoke partnerLink="uIMP110955PL">
      <service name="LVWM_110955:LVWM_110955" port="LVWM_110955HttpEndpoint"/>
    </invoke>
    <invoke partnerLink="uIMPTAlarmPL">
      <service name="LVTAlarm:LVuIMTAlarmProxy" port="LVuIMTAlarmHttpEndpoint"/>
    </invoke>
    <invoke partnerLink="uIMPDryerPL">
      <service name="LVDryer:LVuIMPDryerProxy" port="LVuIMPDryerProxyHttpEndpoint"/>
    </invoke>
  </process>
</deploy>
```

Figure 7.25 BPEL process deployment descriptor file.

7.5.5 **Automated BPEL Process Invocation using ESB based Tasks.**

The implementation of the BPEL process is completed by deploying it on the WSO2 BPS server as a business process service. The BPEL process is can be invoked like a standard web service by the clients and other services. The WSO2 ESB server was used to automatically poll the BPEL business process running on the WSO2 BPS server at regular intervals (5 seconds) using a customised automated task. The ESB uses tasks to invoke business processes hosted on the BPS server. A task in the ESB allows the execution of a piece of

code or a service triggered by a timer. Tasks can be scheduled in the ESB to execute at variable frequency as well as a defined number of times. Figure 7.26 shows a Task called “BPSuIMPWSM” created to invoke the uIMPLVWSMProcess BPEL process at 5 second intervals.

```
<task name="BPSuIMPWSM" class="org.apache.synapse.startup.tasks.MessageInjector" group="synapse.simple.quartz">
   <trigger once="true"/>
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="message">
      <p:uIMPLVWSMProcessRequest xmlns:p="http://polymer.bradford.org/bps/uIMPLVWSM">
         <p:input>?</p:input>
      </p:uIMPLVWSMProcessRequest>
   </property>
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="to" value="http://143.53.207.228:9766/
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="soapAction" value="http://polymer.brad
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="format" value="soap11"/>
</task>
```

Figure 7.26 ESB Task to invoke the BPEL process deployed on the BPS server.

7.5.6 The µIMP Monitor Test bed Layout using Business Processes

The µIMP Monitor project test bed described in section 6.6.4 has been further extended to include BPEL based business process orchestration. This BPEL augmented distributed test bed has been implemented in the centre for Polymer

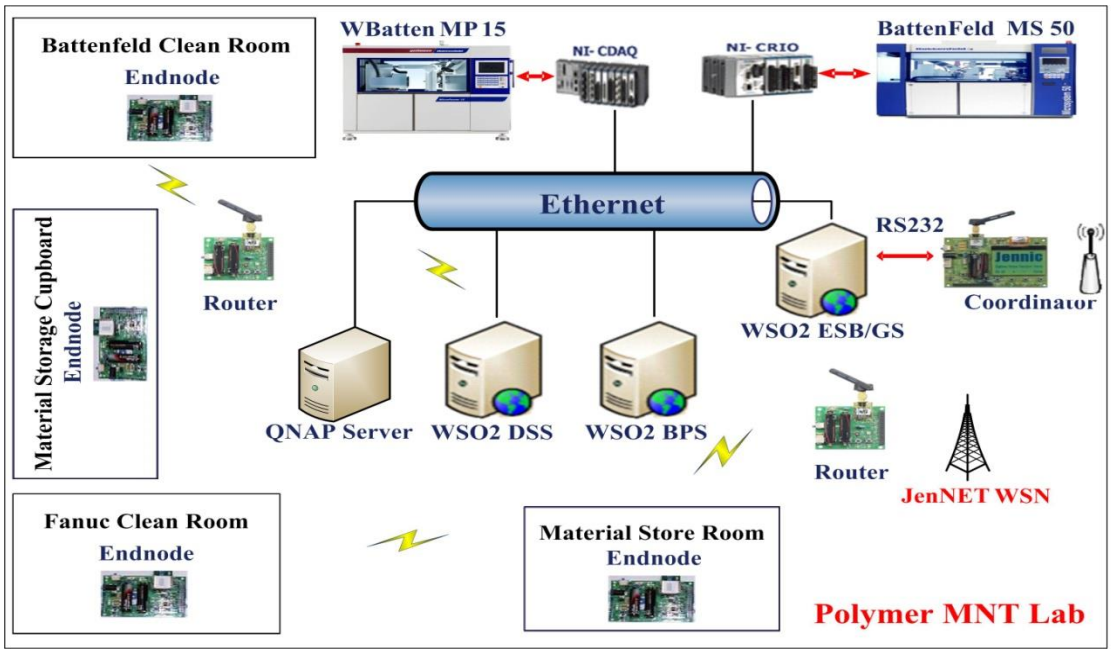


Figure 7.27 Polymer MNT Laboratory Test bed Layout using Business Processes.

MNT at the University of Bradford. A layout of the extended test bed is shown in Figure 7.27. In addition to the hardware described in section, 6.6.4 a distributed instance of the WS02 BPS was setup on a dedicated PC machine. A Jennic WSN JN5148 sensor node placed in the materials cupboard in the Polymer MNT laboratory was used to test the initial framework. The LVWSM process monitoring system was deployed on the same PC machine used by the ESB. The Jennic WSN Coordinator node connected to this machine was used to acquire the sensor readings from the material storage cupboard. The LABVIEW based material dryer and temperature alarm simulations were deployed on the same PC machine used by the WSO2 DSS server.

7.5.7 **Testing**

The initial framework was tested using the BPEL augmented extended architecture. The framework was tested by first initiating the ESB Task "BPSuIMPWSM" which started the "uIMPLVWSMProcess" BPEL process running on the WSO2 BPS server. The WSO2 BPS server logs the number of times the BPEL process completes as well as the time it has been active. Figure 7.28 shows the deployed BPEL business process on the WSO2 BPS server with some statistics on how many times the process was completed and

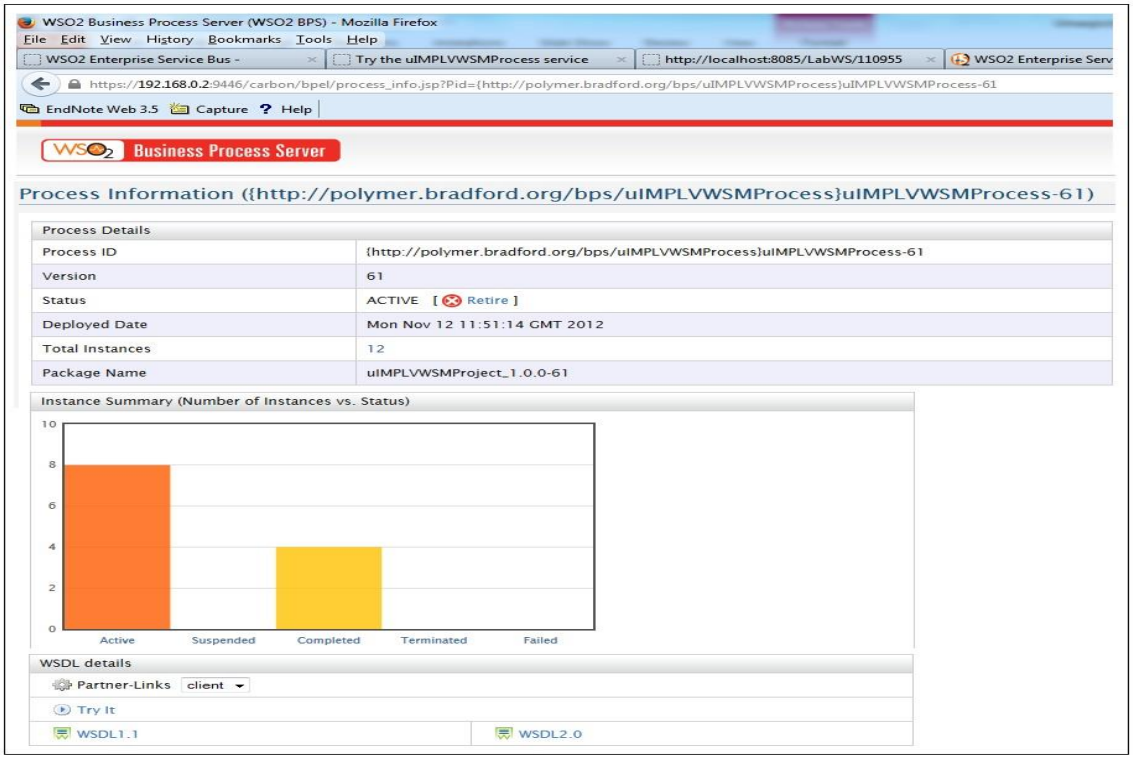


Figure 7.28 uIMPLVWSMProcess BPEL process deployed on the BPS server.

how many instances are active. Next, the BPEL process invocations were tested to see if the services were being invoked correctly. The invocation of the actual RESTful web services running on the LabVIEW Web Application server by the ESB proxy services was tested next. The response from the RESTful web services via the BPEL process was then logged using the main sequence on the WSO2 ESB as well as being graphically displayed in GGs using the WSO2 GS server.

7.5.7.1 *BPEL Process Invocation using ESB Task*

The ESB Task "BPSuIMPWSM" was setup to invoke the BPEL process deployed on the WSO2 BPS server. In this test the BPEL process is invoked by the Task every 5 seconds. A simple switch mediation sequence was created on the ESB to filter responses from the invoked proxy services. If the BPEL process was invoked by the ESB task then a message was logged to indicate that the BPEL process is being invoked. Figure 7.29 shows the part of the output from the ESB mediation log file in which a customized mediation message is being logged indicating that the ESB Task is in the process of invoking the "uIMPWSMProcess" BPEL Process.

```
[2013-02-18 14:26:05,143]  INFO - LogMediator *****MESSAGE***** = *****IN uIMPWSMProcess***** INVOKING uIMPWSMProcess
[2013-02-18 14:26:05,465]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Humidity In LIMITS
[2013-02-18 14:26:10,315]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Humidity In LIMITS
[2013-02-18 14:26:15,444]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Humidity In LIMITS
```

Figure 7.29 Logged Mediator messages on the ESB when invoking the BPEL process.

7.5.7.2 *ESB Proxy Service Invocation from the BPEL Process*

When the BPEL process running on the WSO2 BPS server is invoked by the ESB task, it executes the BPEL workflow shown in Figure 7.23 one step at a time. To verify that the proxy services are being successfully invoked in this workflow the WSO2 TryIt utility was used. The BPEL process returns two separate responses depending upon the received sensor node 110955 readings. If the readings are above the set threshold limits (Humidity > 30% and Temperature > 27C) then the BPEL process returns the Dryer process humidity and the temperature alarm process temperature values. If the sensor readings are below the set threshold values then the process returns the values received

by the sensor node. Figure 7.30 shows the result from two separate invocations. The response on the top shows the result when the sensor readings are above the threshold values. The bottom response shows the result when the sensor readings are within the threshold limits. From this result, we can see that the BPEL process is giving the correct responses based on the sensor readings from the first service it invokes. This result also verifies that the proxy services running on the WSO2 ESB are being invoked in the correct order.

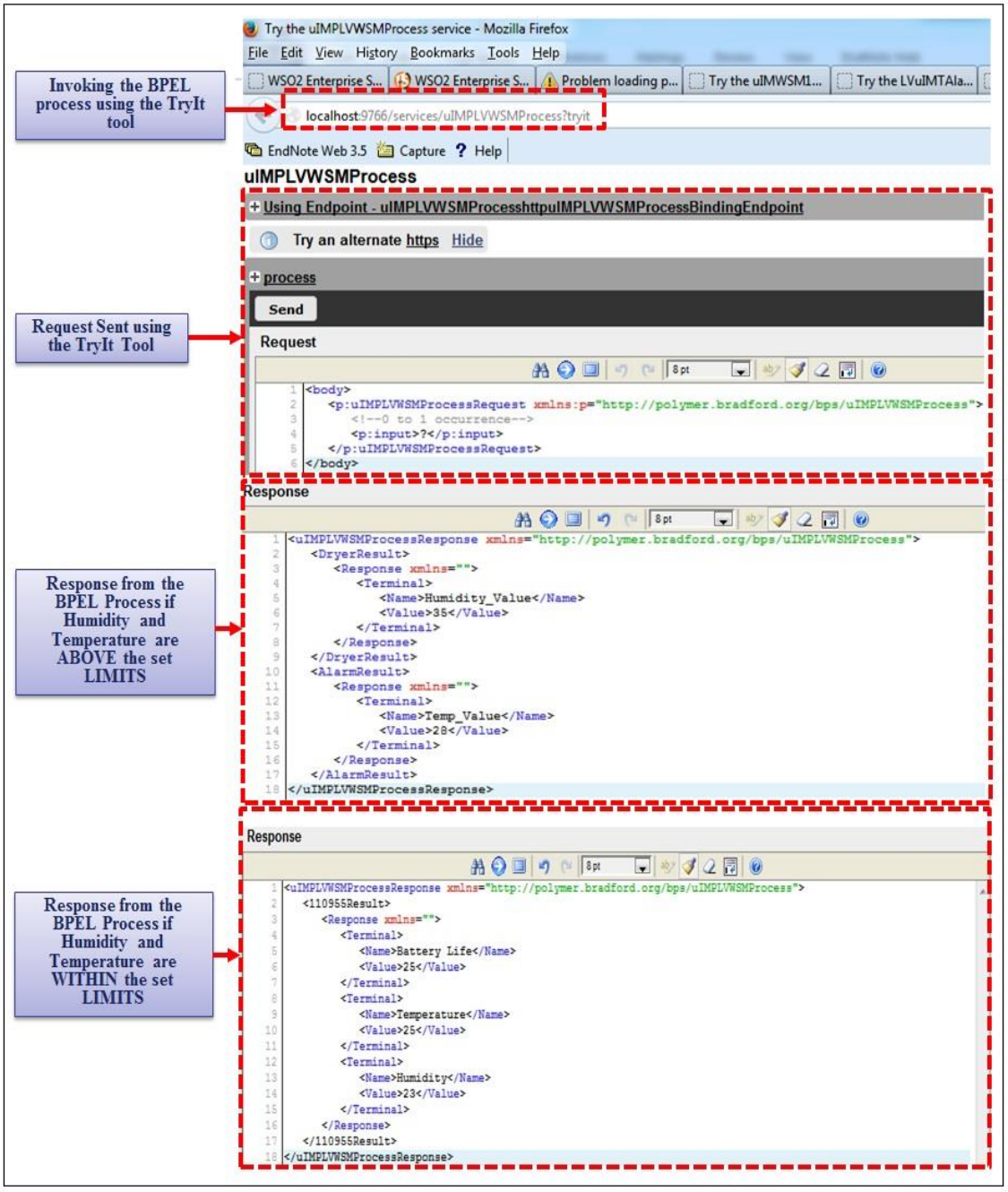


Figure 7.30 BPEL process invocation responses.

7.5.7.3 *RESTful Web Service Invocation by ESB Proxy Services*

The next part of the test was to verify that the proxy services running on the WSO2 ESB are correctly invoking the relevant RESTful web services.

7.5.7.3.1 Sensor Node 110955 Proxy Service Test

The LVWSM process monitoring system creates a RESTful web service called "LabWS" which it deploys on the LabVIEW Application Web Server. The WSO2 TryIt tool was used to invoke this proxy service, which in turn invoked the"LabWS" RESTful web service using the defined endpoint. Figure 7.31 shows the result from invoking the "uIMWSM110955Proxy" proxy service using the TryIt tool. From this result, we can verify that the correct response is being received from the RESTful web service. The response shows the sensor node name, battery life, temperature, and humidity values.

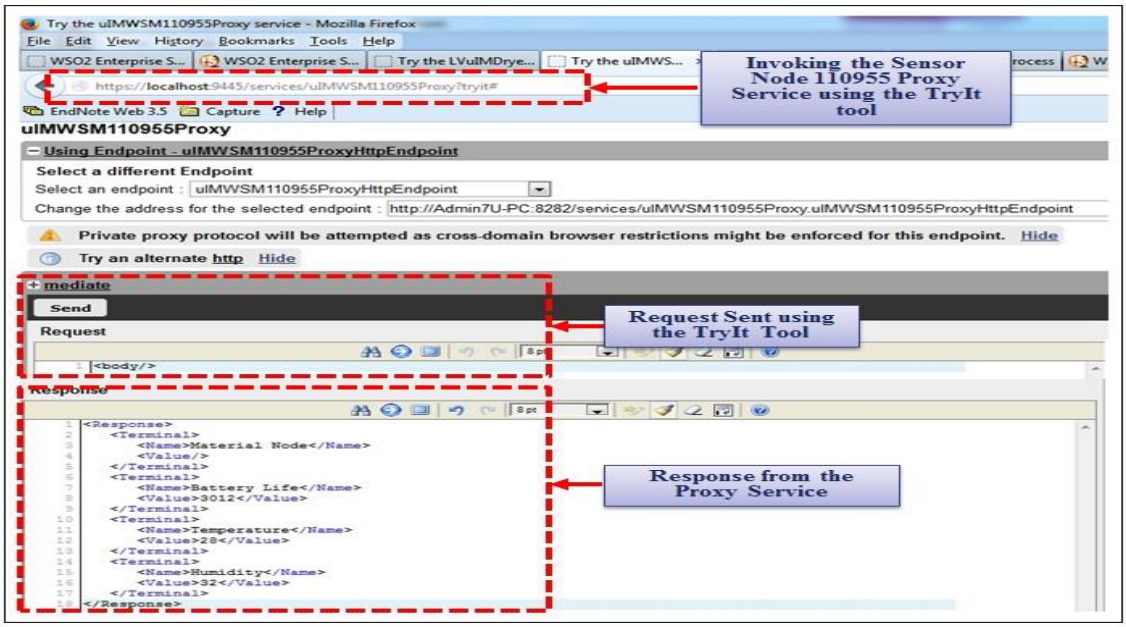


Figure 7.31 Invoking the sensor node 110955 proxy service from the ESB.

7.5.7.3.2 Dryer Process Proxy Service Test

The LABVIEW based "MaterialDryerSim" application creates a RESTful web service called "uIMPBPELDryerWS" which it deploys on the LabVIEW Application Web Server. The WSO2 TryIt tool was used to invoke this proxy service, which in turn invoked the "uIMPBPELDryerWS" RESTful web service using the defined endpoint. Figure 7.32 shows the result from invoking the "LVuIMDryerProxy" proxy service using the TryIt tool. From this result, we can

verify that the correct response is being received from the RESTful web service. The response shows the dryer process humidity value.

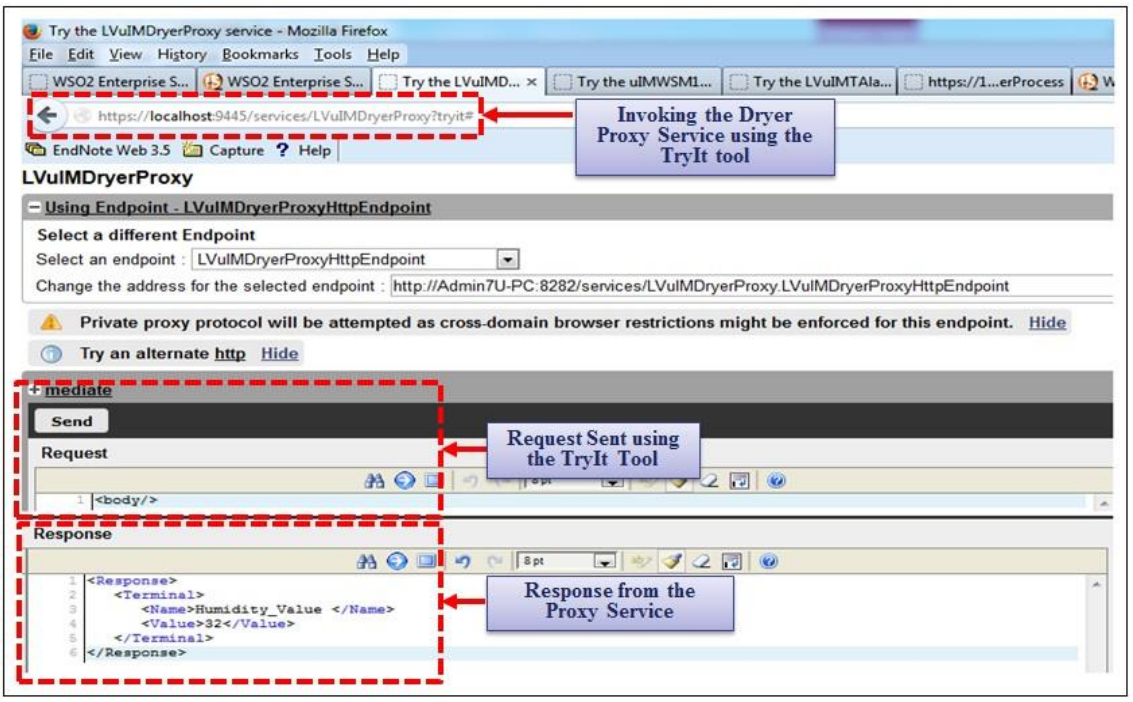


Figure 7.32 Invoking the material dryer process proxy service from the ESB.

7.5.7.3.3 Temperature Alarm Process Proxy Service Test

The LABVIEW based “TemperatureAlarm” application creates a RESTful web

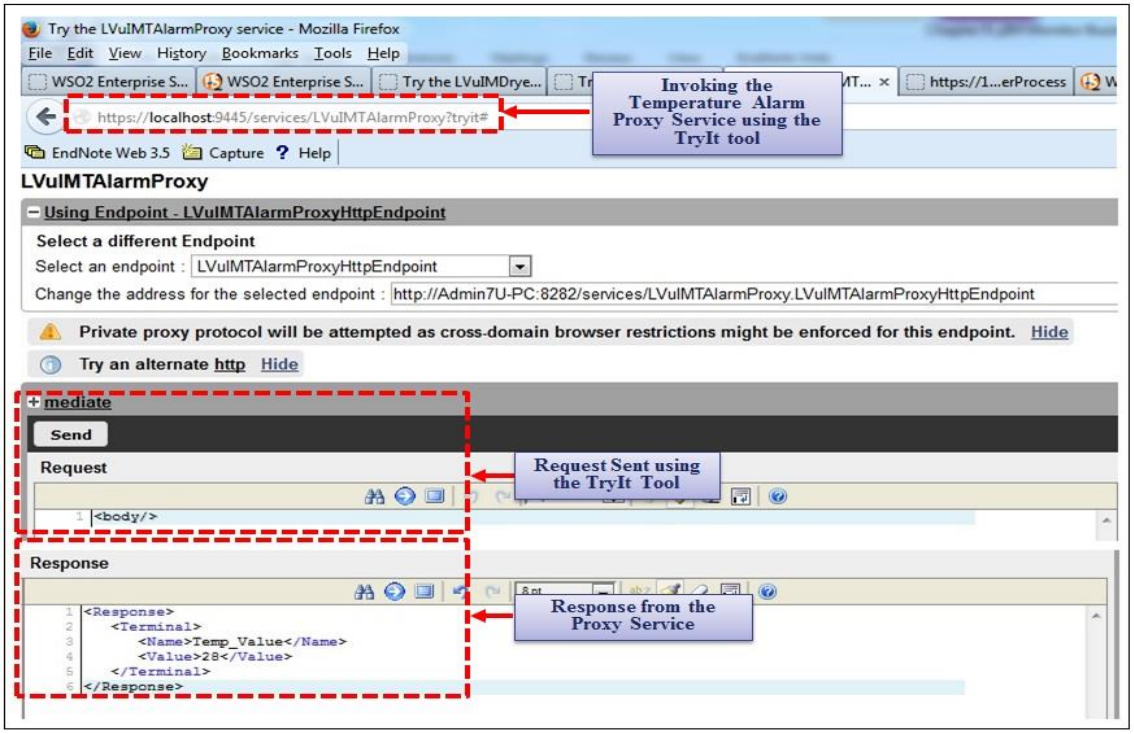


Figure 7.33 Invoking the temperature alarm process proxy service from the ESB.

service called “uIMPBPELTAlarmWS” which it deploys on the LabVIEW Application Web Server. The WSO2 TryIt tool was used to invoke this proxy service, which in turn invoked the “uIMPBPELTAlarmWS” RESTful web service using the defined endpoint. Figure 7.33 shows the result from invoking the “LVuIMTAlarmProxy” proxy service using the TryIt tool. From this result, we can verify that the correct response is being received from the RESTful web service. The response shows the temperature alarm process temperature value.

7.5.7.4 *Displaying the BPEL Process Response*

The final part of the test was to display and verify the response from the BPEL process running on the WSO2 BPS server. This was done in two ways, firstly by logging the response using ESB mediation sequences and secondly by displaying the results graphically using GGs deployed on the WSO2 GG server. The output in Figure 7.34 shows the response being logged into the ESB log file when the material dryer process is being invoked after receiving sensor readings, which are above the threshold limits. Figure 7.35 shows another part of the log file, which shows the temperature alarm process being invoked after receiving sensor readings, which are above the threshold limits. The response from the BPEL process was also sent to the µIMP Web Layer which uses the WSO2 GS server to display the response in a graph format using the GGs API. The WSO2 GS is built upon the WSO2 Carbon platform and uses the GGs API

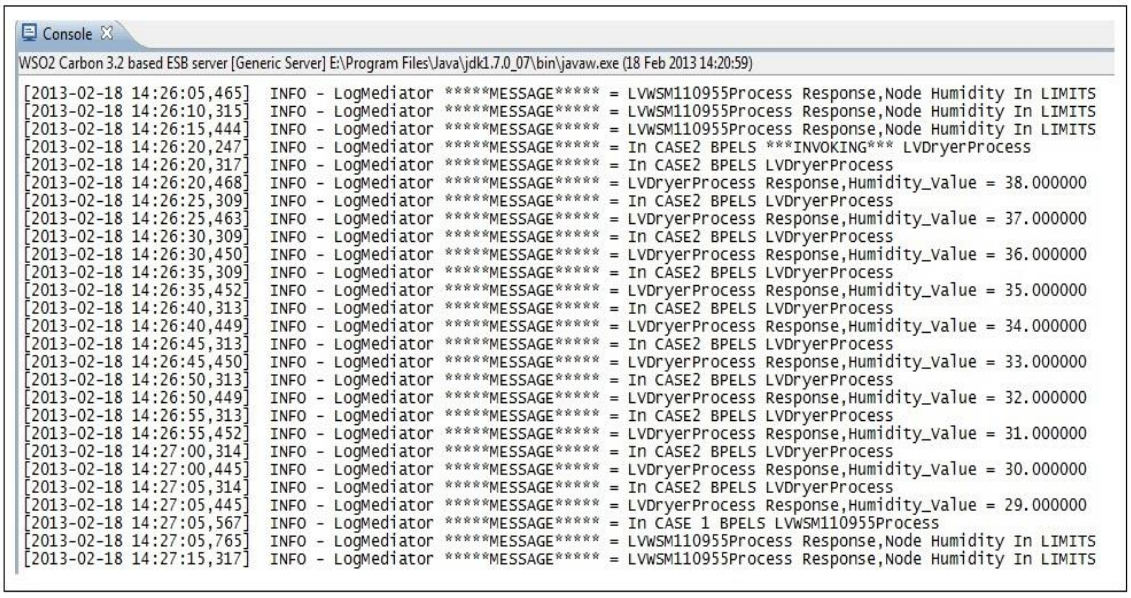


Figure 7.34 Logged Mediator messages on the ESB when invoking the material dryer process.

```
[2013-02-18 14:29:00,210]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Temp In LIMITS
[2013-02-18 14:29:05,432]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS ***INVOKING*** LVAlarmProcess
[2013-02-18 14:29:05,609]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS LVAlarmProcess
[2013-02-18 14:29:05,787]  INFO - LogMediator *****MESSAGE***** = LVAlarmProcess Response,Temp_Value = 28.000000
[2013-02-18 14:29:10,360]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS LVAlarmProcess
[2013-02-18 14:29:10,471]  INFO - LogMediator *****MESSAGE***** = LVAlarmProcess Response,Temp_Value = 28.000000
[2013-02-18 14:29:15,318]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS AlarmProcess
[2013-02-18 14:29:15,433]  INFO - LogMediator *****MESSAGE***** = LVAlarmProcess Response,Temp_Value = 27.000000
[2013-02-18 14:29:20,779]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS AlarmProcess
[2013-02-18 14:29:20,392]  INFO - LogMediator *****MESSAGE***** = LVAlarmProcess Response,Temp_Value = 27.000000
[2013-02-18 14:29:25,269]  INFO - LogMediator *****MESSAGE***** = In CASE3 BPELS AlarmProcess
[2013-02-18 14:29:25,791]  INFO - LogMediator *****MESSAGE***** = LVAlarmProcess Response,Temp_Value = 27.000000
[2013-02-18 14:29:30,277]  INFO - LogMediator *****MESSAGE***** = In CASE 1 BPELS LVWSM110955Process
[2013-02-18 14:29:30,568]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Temp In LIMITS
[2013-02-18 14:29:30,711]  INFO - LogMediator *****MESSAGE***** = LVWSM110955Process Response,Node Temp In LIMITS
```

Figure 7.35 Logged Mediator messages on the ESB when invoking the temperature alarm process.

to create and deploy GGs. Three simple GG applications were developed to display the response from the BPEL process. The GGs poll the BPEL process running on the BPS server at regular intervals (5 seconds) and updates the gadget to display the result in a graph. Figure 7.36 shows the response from the

Figure 7.36 BPEL process response displayed using Google Gadgets.

BPEL process being displayed using two gadgets. The gadget on the top displays the material cupboard humidity whilst the gadget at the bottom shows the response from the dryer process simulation when the humidity goes above the set limit. From these results we can see that the response from the BPEL business process can be logged using the ESB in the mediation log file as well as being displayed in a visual format using the GGs API.

7.6 Conclusions

In this chapter, the μIMP Monitor System presented in chapter six has been extended to incorporate business processes using BPEL. BPEL is used to combine event-driven communication and SOA-based business processes in a μIM environment. Business processes created using BPEL are used to process and analyse the data on a distributed architecture as well as giving the capability to link with other standard business processes. An initial framework for integrating the μIM process with other standard business processes of an enterprise by augmenting the ESB based μIMP Monitor architecture with business processes using BPEL has been successfully implemented and tested. A simple test scenario was implemented in which the environmental conditions of a material storage cupboard were used as the trigger to initiate two simulated processes. The humidity and temperature conditions in the material storage cupboard were monitored using a WSN. A BPEL process used the humidity reading from the storage cupboard to trigger a material drying process if the humidity level was above a set threshold. The temperature reading from the material cupboard was used to trigger a separate temperature alarm if the temperature was above a set threshold.

The results from the implementation and testing of the framework show that by using sensor technology and web services the functionality of a typical business process on the factory shop floor can depicted. A WSN was used as the sensor technology and web services (Restful and SOAP) along with BPEL services as the glue, which linked the sensor technology to simulated processes. It has been shown through the framework that disparate environments based on various sensor technologies could be linked using BPEL. Although only a single sensor technology was implemented and tested with simulated environments, future work can look at integrating the distinct high-resolution machine data with the environmental data using this framework and other sensor technology. Finally, it has been shown that disparate environments based on distinct and unique sensor technologies can be linked to form a cohesive business process

Chapter 8: Conclusion and Future Work

Up until now the main focus of WSN and the Internet of Things (IoT) concept has been on non-critical home and office environments. In these environments, the interoperability and data fusion between earlier Enterprise-IT applications and other applications was difficult due to their proprietary nature. The adoption of the SOA approach saw the elimination of these problems in enterprise systems. The Enterprise Service Bus (ESB) has been used as the enabling middleware technology for implementing the SOA. The ESB has allowed a very large number of services to communicate with each other by using messaging and asynchronous communication to remove the integration complexity from the different systems involved. Furthermore, most of the latest enterprise applications are increasingly built around the SOA architecture, which has enabled the seamless integration of disparate information systems from different vendors. However, the factory shop floor in industrial environments is still at an early stage of implementing the SOA.

Some factory shop floor industrial environments have seen the convergence, (from advances in computing and communication capabilities), in devices and equipment with the web using the SOA, increasing the use of WSN and the IOT. In these environments, the main focus of the implementation of the SOA paradigm has been the WSNs through web services at both the device and gateway levels. The issues such as interoperability, sensor fusion, and data semantics amongst the disparate devices and equipment on the shop floor still pose a huge challenge. If SOA is implemented on all the devices and equipment on the factory shop floor, then this will result in a large number of permutations services and composite services. These services combined with other business level components can pose a huge challenge to manage, as it is often difficult to keep an overview of all the devices, equipment and services. Hence, there is a need to integrate the heterogeneous devices and the arising complexity seamlessly under one platform (e.g. ESB). Furthermore, the plastics industry environment made up of the Injection Moulding (IM), Micro Injection Moulding (μIM) and Extrusion line manufacturing processes is one industrial environment, which still has not had much exposure to WSNs and the IOT concept. These processes would directly benefit from WSN and the IOT

concept by making the environmental and high-resolution process data available in real-time over the internet. The main contributions of this thesis are summarised in the following sub sections followed by a discussion of future research directions.

8.1 Selecting the WSN for the µIM Process Monitoring

Before a process monitoring system based on WSN could be developed to monitor the µIM process, a WSN was selected from a wide range of vendors, which could meet the required monitoring specifications.

In Chapter 3, the most common parameters used for monitoring and defining the plastics machine operating environment are identified. Based on these parameters, three WSN kits were evaluated from nine kits for the usability and reliability of the accompanying tools, and reliability in a plastics industry setting. As to date, there was no comprehensive review of different WSN kits being tested in industrial settings. The SUNSPOT kit was a strong candidate for this project due to its high processing power (> 180MHz), memory and software development tools. However, out of the final three kits evaluated, the SUNSPOT kit was found to be the most unreliable. It lacked support from its manufacturer as well as problems of random loss of connectivity during the running of applications and updating of the firmware on the nodes. Therefore, due to these identified problems and the lack of future support, further testing of the SUNSPOT kit was deemed to be not useful and discontinued.

The XBee kit had comprehensive support from Digi, as it was a well-developed and supported product with a number of different versions of the radio. However, it did have some shortcomings such as the lack of complex examples on how to integrate the XBee devices in applications using a fully developed SDK. . The second problem was that the XBee devices were used on their own without a dedicated micro-controller board. The on-board Freescale 16bit HCS08 micro-controller and memory were dedicated for the functionality of the RF device through AT modem or API commands. This meant that in order to have a complete sensor node, which can be programmed using python IDE hence allowing on-board processing of data; the XBee devices had

to be interfaced with a separate dedicated micro-controller platform such as the Arduino boards.

The Jennic kit on the other hand came with a fully developed SDK as well as a fully developed tool-chain. The Jennic WSN kit and the accompanying tool-chain were very reliable and easy to use. The hardware as well as the software provided by Jennic was excellent in terms of reliability and ease of use. Jennic not only supports the IEEE 802.15.4 and ZigBee protocols but they also have their own protocol stack called JenNET, which allows for the creation of large networks with up to 500 nodes.

In Chapter 4 the XBee and Jennic kits are further evaluated for data reliability using the PER and RSSI/LQI parameters in the home/office and polymer industrial environments. Although the PER, RSSI, and LQI parameters had been used in a number of studies to assess the communication reliability of WSN in the presence of interferences from different sources. As to date there was no study using these parameters to evaluate and compare the reliability and performance of Jennic and XBee WSNs in the home/office and industrial environments. A comprehensive analysis of the RSSI and PER using Jennic and XBee modules in two distinct environments was carried out. The results showed that the XBee modules performed slightly better in RSS and PER in the polymer industrial environment. Also taking into consideration that in these tests the number of data packets analysed for the Jennic modules for each test was 50000 packets per test whereas for the XBee modules it was 500 packets per test. This was purely due to the restrictions in the software used. Therefore based on the worst case scenario results using the standard three retries the highest PER for the XBee modules was 0.4% and that for the Jennic modules was 0.475%. Based on these results there is not much difference in terms of PER between the two modules and communication reliability can be considered reliable with these results. The fact that for the Jennic modules, 50000 packets were analysed for each test and the PER is below 0.5%, made them the primary choice for monitoring the polymer industrial environment. Based on this conclusion the Jennic kit was chosen to be used for the monitoring of the polymer industrial environment.

8.2 LVWSM: WSN based µIM Process Monitoring System

All the previous research focus was on the monitoring of the IM/µIM processes, resulting in a number of customised process monitoring systems, which were proprietary in nature and difficult to integrate or interoperate with other systems and business components of the enterprise. In chapter 5 the complete architecture, design, development, and implementation of the LVWSM µIM process monitoring system was presented which tried to address the above integration issue through the use of WSN and Web Services. The chapter focused on two distinct parts of the system: WSN hardware programming and interface; and the LVWSM µIM process monitoring application.

The firmware for the Jennic JN5148 sensor nodes was modified and reprogrammed. To monitor the µIM machine process parameters, a state machine was implemented to read the analogue ports and convert the analogue signal into digital form using the on-board ADC. A hardware interface was developed to connect the wireless sensor nodes to the µIM machine. The initial test using terminal software showed that the data from the WSN was being sent back to the gateway laptop in the correct format. The machine sensor outputs were being converted in digital format using the programmed state machine. To address the dynamic nature of the WSN such as nodes joining and leaving; transmitting live data from various locations; required the creation of a novel application, which could consider these requirements. The novel LVWSM architecture used the concept of dynamic application instance using application templates. This had further challenges of sorting the dynamic data coming from various sensor nodes at the same time. A novel data parser was developed which used data token strings to overcome this challenge. Two types of sensor node templates were developed to cater for the different types of nodes. The use of dynamic instances and templates made this architecture scalable hence making the integration of WSN from other manufacturers easier in the future. The Sensor Web layer was developed to allow the data from the µIM process to be converted into Web Services allowing integration with other web applications and components of an enterprise.

The testing of each layer was done separately and the complete prototype system was further tested in a number of scenarios. Results from these tests

showed that the LVWSM µIM process monitoring system had the ability to detect dynamic nature of the nodes as well as displaying the live data from the process with options of viewing individual sensor data. The data could also be logged by the software in a file format as well as in a relational database. During the testing of the complete software a number of minor issues were detected and rectified. One key issue that became known with this system was the resolution of the machine data was low around an average of 15Hz. This resolution will be fine for monitoring the environemntal variables but it will not be sufficient for the kind of resolution required when monitoring high pressures and temperatures for the purpose of process characterisation. The other problem which came to light was that the WSN nodes could only send a maximum of approximately 250Kb per second in theory but in reality this was closer to 120Kb. Therefore when dealing with high-resolution data this would be a problem and would require a higher resolution ADC and data storage on the sensor node. To solve this issue a number of other options were available; one was to use an alternative WSN which can cater for the above shortcomings or the other option is to use an altenative data acqusition technology such as Compact-RIO from national instruments for high-resolution machine data. The latter option was chosen as the former option required the development of a new hardware which was outside of the scope of this thesis. The Compact-RIO option meant the development of a novel hybrid architecture which would allow both the current WSN based LVWSM system and a high-resolution DAQ system to coexist and interoperate.

8.3 µIMP Monitor: An SOA based Hardware Assimilation Architecture

In order to allow the WSN based LVWSM system to coexist and interoperate with other disparate systems, chapter 6 proposed the µIMP Monitor: a novel four layer hardware assimilation architecture with the ESB at its core for the purpose of integrating high-resolution µIM machine process data and low-resolution WSN (environmental and business process) data. This novel assimilation architecture can be used as a generic approach to integrate WSN from different vendors and disparate hardware in application environments other than the polymer industrial environment. The ESB was used as the

enabling middleware technology for implementing the SOA hence removing the integration complexity from the hardware systems. The architecture was made up of four clearly defined inter-related layers:

µIMP Hardware Layer: This layer consists of various hardware platforms, which are connected to the relevant applications in the µIMP Local Monitor Layer. Depending upon the hardware used and how the SOA paradigm has been deployed on the actual hardware (see section 2.6.2) this layer makes available the device level services or raw data to be used by the µIMP Local Monitor layer applications. In this thesis, we have deployed the SOA at the gateway middleware hence resulting in raw sensor data.

µIMP Local Layer: This layer was responsible for exposing the raw data (recorded using the various hardware platforms) using local applications. The local applications in this layer use a common base architecture, which allows the formatting of the raw sensor data into file and database formats as well as converting it in Restful Web Services. Each application in this layer creates services that support multiple message formats (JSON, XML, HTML, and Text) and transport protocols (HTTP, JMS). In this thesis, XML messages over HTTP/HTTPS transport have been implemented.

µIMP Global Layer: The size of the factory shop floor will determine the number of web services being created by the local applications. Large number of web services can become difficult to manage and hence require web service enabled composition technology such as an ESB to manage it globally. Using the ESB at its core, this layer was designed to acts as the common data communication and messaging layer between the different local applications and frontend web applications. It consists of four main components; the Data Service Server (DSS), the ESB, the Web Interface, and the Service registry and has four main functions: data management, application management, service management, and user authentication and authorisation. The orchestration of the business processes using BPEL is initiated in this layer.

µIMP Web Layer: This layer was designed to allow the monitoring of the µIM process on the web. Although the GG API was implemented in this thesis, various other web-based applications, which consume web services, can be

easily integrated in this architecture. The Google Gadgets use JavaScript and XML/HTML to communicate with GG API. One of the advantages of using Google Gadgets is that mini applications can be developed for the different stages of the µIM process. Linking these applications allows the development of complex applications, which are modular in nature. The WSO2 GS was implemented as the frontend web application of µIMP Monitor architecture and allows the monitoring of the µIM process on the internet using Google Gadgets.

The proposed architecture was implemented and tested on a distributed experimental server test bed using the LVWSM µIM process monitoring system described in chapter 5; The LVSimulator applications for generating machine data; the WSO2 SOA suite of tools and the GG API. The LVSimulator applications were primarily used for testing the architecture without connecting the actual hardware. The Web Services from these simulator applications along with the Web Services created by the LVWSM µIM process monitoring system were used to test and evaluate this architecture. A number of test scenarios were implemented on this architecture.

Simulated machine data generated by the LVSimulator applications was used to demonstrate the integration of the proposed test bed environment. The simulated data was converted into a RESTful web services and visualized on the WS02 GG server as gadgets. On the ESB, proxy services were created and exposed through endpoints pointing to the LABVIEW RESTful web services. A number of Google Gadgets were developed which send XML requests to the ESB at regular intervals and visualize the result if an update is available.

The simulated data was then replaced with LVWSM µIM process monitoring system which uses the Jennic WSN. The sensor nodes in the Jennic WSN have integrated Sensirion SHT11 single-chip multi-sensor module for the measurement of temperature and relative humidity. The two sensor readings and node voltage reading were extracted and converted into a RESTful web service. On the ESB, proxy services were created and exposed through endpoints. A number of Google Gadgets were created to visualize the results in different formats.

The LVWSM µIM process monitoring system and the LVSimulator applications were then all connected simultaneously. On the ESB, proxy services were created and exposed through endpoints. A number of Google Gadgets were created for all the connected systems and visualised under a single unified web portal.

Experimental results from these test scenarios showed that the architecture was highly scalable and could allow potentially a large number of disparate hardware systems to be connected and visualised under a single unified web portal. The proposed ESB with its graphical user interfaces proved to be very flexible and easy to configure. It was highly scalable with a potential of hosting a very large number of services at any given time, therefore making it a suitable composition methodology for integrating large number of devices on the shop floor. The use of Google Gadgets API to develop mini applications for the different stages of the µIM process was feasible hence paving the way for future development of complex web-based process monitoring applications, which are modular in nature.

8.4 µIMP Monitor: A Business Processes Framework

Chapter 7 proposed a new framework for integrating the µIM process with other standard business processes of an enterprise by augmenting the ESB based µIMP Monitor architecture with business processes using BPEL. In this framework, the functionality of each stage in the process is abstracted using a relevant sensor technology and Web Services. For each stage, the services can be made up of composite services, which can be any process parameters like the temperature, pressure, or screw position provided by the sensor technology. The composite services could be made up of further services for example screw position can have the velocity and displacement services. Similarly, the standard business processes can be made up of composite services, which depict the behaviour and functionality of the underlying process. The services and composite services in each stage can be linked with each other using BPEL, which acts as the “glue” to bind these Web Services (depicting the functional behaviour of smaller business processes) in each stage hence allowing the formation of a single unified business process. The composition,

binding and orchestration of services in order to fulfil a processes needs allows the formation of a loosely coupled process monitoring system.

Business processes created using BPEL were implemented on the extended distributed experimental testbed to allow the processing and analysis of the data as well as giving the capability to link with other standard business processes. An initial framework was successfully implemented and tested for integrating the µIM process with other standard business processes of an enterprise. A simple test scenario was implemented in which the environmental conditions of a material storage cupboard were used as the trigger to initiate two simulated processes. The humidity and temperature conditions in the material storage cupboard were monitored using the LVWSM µIM process monitoring system. A BPEL process used the humidity reading from the storage cupboard to trigger a material drying process if the humidity level was above a set threshold. The temperature reading from the material cupboard was used to trigger a separate temperature alarm if the temperature was above a set threshold.

The results from the implementation and testing of the framework show that by using sensor technology and web services we can depict the functionality of a typical business process on the factory shop floor. We used the WSN as the sensor technology and web services (Restful and SOAP) along with BPEL services as the glue, which linked the sensor technology to simulated processes. It was shown through this framework that disparate environments based on distinct and unique sensor technologies could be linked to form a cohesive business process.

This research has allowed for the use of the SOA with the ESB at its core and a WSN to measure and visualize the µIM process. However to reach this point, it underwent a three stage process; a) the initial selection criteria gave a more focused approach and inevitably lead to a more (stage b and c) detailed problem solution process. The selection of a WSN for the µIM environment was made by gathering all the information from nine different kits (tested for use in previous research and the integrated development environment for operability). Based on this, the kits for future use were narrowed down to two. One of the

further main selection criteria was reliability in the polymer industrial environment, which allowed the narrowing in kit selection down to one. The second stage proposed the LVWSM, a novel µIM process monitoring system which gave rise to the following issues; nodes joining and leaving the network dynamically (solved using application templates), nodes were transmitting the data simultaneously (solved with data parser using data token strings), interoperability with other applications and enterprise components (solved through the Sensor Web layer using Web Services). The results from this stage indicated that the WSN is suitable for monitoring low-resolution environmental data but not suitable for monitoring high-resolution machine data and hence requires a new integration approach, which lead to the third stage. The third stage proposed a novel integration architecture µIMP Monitor for monitoring the µIM process, which solved the following issues:

a) High-resolution machine data (solved through the µIMP Global Layer using the ESB).
b) Interoperability of high-resolution µIM machine process data and low-resolution WSN (solved using Web Services managed via the ESB).
c) Visualising the different stages of the µIM process (solved through the µIMP Web layer through the use of the Google Gadgets based WSO2 Viskit API).
d) Integrating the µIM process with other standard business processes of an enterprise (solved using WS-BPEL based business processes framework).

As this is the first time such research is being carried out, both solutions and problems were found as the project commenced. It has been possible for the first time to use the SOA and WSNs to measure, visualise and orchestrate the µIM process, which has now opened up the possibility for a broad range of future work.

8.5 Future Work

The research carried out in this thesis provides a way to implement the IOT concept using WSN in an industrial environment. The µIM process was tackled by using; monitoring systems, simulation applications, Google Gadget applications, ESB based assimilation architecture and business processes

framework hence providing a test bed for future research. While much has been achieved during this study, several problems remain to be resolved and investigated for future research:

8.5.1 Mathematical Model for the dynamic nature of the environments

In order to more accurately study the effects of industrial environments on WSN it would be beneficial to model the dynamic nature of the industrial environment. A mathematical equation based on the results obtained for RSSI and PER in real environments should be developed which takes into account the dynamic factors and parameters of these industrial environment.

8.5.2 Further Develop the LWSM µIM process monitoring system

In order to develop the LVWSM µIM process monitoring system into a viable commercial product further research could be carried out for the integration of; intelligent data analysis techniques, sensor fusion algorithms and other systems as an instance of the network.

Integrate intelligent data analysis techniques: The data gathered from the environment is transmitted wirelessly to a central gateway, which extracts and classifies the data according to the type of node being used. It then formats and presents the data to be logged into a file or a relational database as well as being converted to a service through the use of Restful Web Services. Future work in this area could look at using intelligent data mining to identify process trends, variations in material and product quality. The use of Artificial Intelligence (AI) techniques such as Genetic Algorithms (GA) (Pinto et al., 2014), Neural Networks (NN) (Safari et al., 2014), or software agents (Wang et al., 2014) should be investigated for automatic optimisation and intelligent control of the process.

Integrate Sensor fusion algorithms: The large amounts of data gathered from various disparate sensor sources (e.g. high-resolution machine data and environmental data), can become very difficult to; read, understand and use for a meaningful purpose. For this reason, further research into sensor fusion is required. This would mean the combining of sensory data or data derived from sensory data in such a way that the resulting information is in some way better to understand and use for a meaningful purpose. There are a number of

techniques which could be used for this purpose such as Kalman filters, NNs, Software Agents, GA etc (Wang et al., 2014, Safari et al., 2014).

Integrate other systems as an instance of the network: The LVWSM architecture used the concept of dynamic application instance using application templates. Two types of sensor node templates were developed to cater for the different types of nodes. Future work into the development of the LVWSM architecture to make it more generic and hence allowing the integration of WSNs from other manufacturers would be highly beneficial. This could be then extended to include hybrid wireless/Ethernet high-resolution machine data systems such as the NI-CRIO system. Follow on from this would look at how devices on a hybrid network could be treated like sensor nodes on a WSN using Cyber Physical Systems approach, which can join and leave the network dynamically. The effect of high-resolution data on the network bandwidth could be investigated and how this would have an impact on the real timeliness of the data.

8.5.3 Further develop the µIMP Monitor Architecture

The future work the µIMP Monitor architecture needs to be extended in a number of areas. Further research could be carried out into the following areas (including; testing with high-resolution machine data, processing of high-resolution data, virtualization and µIMP Web layer improvements):

Test with high-resolution machine data: The ESB based µIMP Monitor architecture has been tested using WSN and simulated machine data. The future work needs to test the architecture using high-resolution machine data. This would require the further development and completion of the NI Compact-RIO based LVCRA system and the NI CDAQ based LVCDA system. To improve the reliability and real timeliness of the architecture alternative transport protocols such as JMS, RTP, XMPP (Apache, 2012d) need to be investigated. The architecture could also be further evaluated in terms of data size and parsing time by using alternative messaging (JSON, Text, HL7) (Jayasumana, 2012) protocols.

Data Mining, Semantics and Fusion for high-resolution data: The process data from the µIM machines is recorded using NI DAQ devices at acquisition rates in excess of 1000Hz. This high-resolution data transmitted over the local ether network can have a huge impact on the network bandwidth. For such data to have significant value, it should provide an accurate representation of the actual process conditions at the time the record was made and that potential sources of error are identified and mitigated. Therefore, there is a need for on node data processing algorithms and techniques. Data mining, data semantic and AI techniques (Machine learning, NNs, GA) need to be further investigated for the purpose of extracting meaningful data (trend analysis, key process parameters) from a large set of process data in real-time. Furthermore the NI-Compact-RIO system has an on board field-programmable gate array (FPGA). Further research is needed into how this could be best utilised for the purpose of sensor-level signal conditioning, inline filtering, high-speed data collection and analysis, and mission-critical safety logic.

Further research is also required to monitor the machine data on the web. Various studies have been carried out using web mining and semantics to analyse unstructured data by adding semantic annotations in order to access meaningful data (Rettinger et al., 2012). Furthermore web semantics have also been combined with AI techniques such as software agents (Singh, 2012) to extract knowledge from an unstructured set of data. Further research could look at storing and processing the industrial environment data in a cloud setup and applying the web semantics and AI techniques in the cloud. This topic has its own set of issues and challenges such as security, data processing, bandwidth, network connection, third party service providers to name a few.

Virtualisation Intelligence and Generic Architecture development: The recent years has seen the proliferation of the virtualization and cloud computing technologies in various fields. These technologies allow the management of system functionality and resources regardless of their physical locations and are changing the way businesses and users interact with IT resources (Pei-Breivold et al., 2013). The industrial environments such as the polymer machinery environment have still not been exposed to these technologies. Although some aspects of virtualisation were explored and tested in this thesis, future research

could focus on the complete µIMP Monitor distributed server architecture to be virtualised on one powerful host server.

Another future work direction could possibly look at improving the µIMP Monitor architecture by making it more intelligent using existing SOA products from WSO2. This could potentially be done by augmenting the µIMP Global layer with the complex event, business rules, and task servers. These servers if combined together could significantly improve the architecture in terms of data analysis and automation. The architecture could be further improved by making it more generic so that it can be used not only in the polymer industrial environment but also in other industrial and home/office environments.

µIMP Web layer improvements: Another future direction could look at investigating into alternative web technologies to the Google Gadgets API. Alternatives to the WSO2 web visualisation libraries could also be explored and evaluated in this layer. Further research into inter-gadget communication techniques with integrated intelligence could be useful for the creation of complex web applications intelligent web analytics capability.

8.5.4 Further develop the Business Processes Framework:

The framework has been evaluated using a couple of business process scenarios. Future work could include the implementation and evaluation of more complex business process scenarios. This could be then extended by look at implementing a complete product life cycle from the point of material delivery to the end packaged product delivered to the customer. This would require research into modelling the various other business processes of an enterprise using this framework other sensor technologies. In this thesis, we have only implemented a single sensor technology and tested it with simulated environments. This could be expanded as part of the future work to look at integrating the high-resolution machine data (using the NI Compact-RIO technology), environmental data (using the WSN) and non-industrial processes data such as purchasing and packaging (using RFID) for the purpose of material and product monitoring

.

References

Akyildiz, I. F., Weilian, S., Sankarasubramaniam, Y. and Cayirci, E. (2002) A survey on sensor networks. *Communications Magazine, IEEE,* 40 (8), pp. 102-114.

Amini, N., Wenyao, X., Zhinan, L., Ming-Chun, H. and Sarrafzadeh, M. (2011) Experimental analysis of IEEE 802.15.4 for on/off body communications. In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on.* pp. 2138-2142.

Apache (2012a) *Apache Qpid.* Code. Available from: http://qpid.apache.org/

Apache, S. F. (2012b) *Apache Axis2 - Apache Axis2/Java - Next Generation Web Services.* Code. Available from: http://axis.apache.org/axis2/java/core/

Apache, S. F. (2012c) *Apache Orchestration Director Engine (ODE) – Apache ODE™.* Available from: http://ode.apache.org/index.html (Accessed 11/10/2012).

Apache, S. F. (2012d) *Apache Synapse - The Lightweight ESB.* Repository. Available from: http://synapse.apache.org/

Arduino (2013) *Arduino - HomePage.* Available from: http://arduino.cc/en/ (Accessed 16/12/2012).

Aripin, N. M., Fisal, N. and Rashid, R. A. (2009) Performance Evaluation of Video Transmission Over Ultrawideband WPAN. In: *Modelling & Simulation, 2009. AMS '09. Third Asia International Conference on.* pp. 676-680.

Assous, N., Lebedev, N., Daviot, R. and Abouchi, N. (2009) Wireless Sensors for Instrumented Machines: Propagation Study for Stationary Industrial Environments. In: *Computer Aided Modeling and Design of Communication Links and Networks, 2009. CAMAD '09. IEEE 14th International Workshop on.* pp. 1-5.

Bertocco, M., Gamba, G., Sona, A. and Vitturi, S. (2008) Experimental Characterization of Wireless Sensor Networks for Industrial Applications. *Instrumentation and Measurement, IEEE Transactions on,* 57 (8), pp. 1537-1546.

Bradford University (2010) *The Centre for Polymer Micro and Nano Technology.* Available from: http://www.polymer-mnt.brad.ac.uk/ (Accessed 12/08/2010).

Candido, G., Sousa, C., Di Orio, G., Barata, J. and Colombo, A. W. (2013) Enhancing device exchange agility in Service-oriented industrial automation. In: *Industrial Electronics (ISIE), 2013 IEEE International Symposium on.* pp. 1-6.

Caro, D. (2008) *Wireless Networks for Industrial Automation* 3rd ed. ISA.

CAS (2012) *OPC Unified Architecture.* Available from: http://www.commsvr.com/UAModelDesigner/Index.aspx

Cena, G., Bertolotti, I. C., Valenzano, A. and Zunino, C. (2007a) Reasoning about communication latencies in real WLANs. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on.* pp. 187-194.

Cena, G., Bertolotti, I. C., Valenzano, A. and Zunino, C. (2008a) Industrial applications of IEEE 802.11e WLANs. In: *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on.* pp. 129-138.

Cena, G., Valenzano, A. and Vitturi, S. (2007b) Wireless Extensions of Wired Industrial Communications Networks. In: *Industrial Informatics, 2007 5th IEEE International Conference on.* Vol. 1, pp. 273-278.

Cena, G., Valenzano, A. and Vitturi, S. (2008b) Hybrid wired/wireless networks for real-time communications. *Industrial Electronics Magazine, IEEE,* 2 (1), pp. 8-20.

Chao Hu, Z., Liu Yingzi, P., Zhenxing, Z. and Meng, M. Q. H. (2009) A novel FPGA-based wireless vision sensor node. In: *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on.* pp. 841-846.

Chen, D., Nixon, M., Aneweer, T., Shepard, R. and Mok, A. K. (2004) Middleware for wireless process control systems. In: *Architectures for Cooperative Embedded Real-Time Systems Workshop.*

Chen, L. and Lu, X. (2009) Achieving Business Agility by Integrating SOA and BPM Technology. In: *Information Technology and Applications, 2009. IFITA '09. International Forum on.* Vol. 1, pp. 334-337.

Chen, W.-C., Fu, G.-L., Tai, P.-H. and Deng, W.-J. (2009) Process parameter optimization for MIMO plastic injection molding via soft computing. *Expert Systems with Applications,* 36 (2), pp. 1114-1122.

Chenhui, X. and Xinran, L. (2012) General analysis on architecture and key technologies about Internet of Things. In: *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on.* pp. 325-328.

Chilo, J., Karlsson, C., Angskog, P. and Stenumgaard, P. (2009) EMI disruptive effect on wireless industrial communication systems in a paper plant. In: *Electromagnetic Compatibility, 2009. EMC 2009. IEEE International Symposium on.* pp. 221-224.

Chuan-Yu, C., Cheng-Wei, L. and Jia-Shung, W. (2008) Reliable grouping GAF algorithm using hexagonal virtual cell structure. In: *Sensing Technology, 2008. ICST 2008. 3rd International Conference on.* pp. 600-603.

Chun, Y., Fang, Y., Qian, W. and Nai-jia, L. (2012) Research and application on university data sharing and exchanging. In: *Information Technology in Medicine and Education (ITME), 2012 International Symposium on.* Vol. 1, pp. 220-223.

Cobban, M. (2004) *What is BPEL and why is it so important to my business?* Available from: http://www.softcare.com/whitepapers/wp_whatis_bpel.php (Accessed 14/09/2010).

ConnectPort (2012) *ConnectPort X4/X4 H - 2G/3G/4G Customizable Commercial Grade Routing Gateway - Digi International.* Available from: http://www.digi.com/products/wireless-routers-gateways/routing-gateways/connectportx4 (Accessed 6/11/2012).

Davenport, T. H. and Prusak, L. (1998) *How Organizations Manage What They Know.* Business School Press.

de Souza, L. M. S. a., Spiess, P., Guinard, D., K\"o, h. M., Karnouskos, S. and Savio, D. (2008) SOCRADES: A Web Service Based Shop Floor Integration Infrastructure. In: *Internet of Things 2008 Conference, Zurich, Switzerland.* Springer, Vol. 4952, pp. 50--67.

Decotignie, J. D. (2005) Ethernet-Based Real-Time and Industrial Communications. *Proceedings of the IEEE,* 93 (6), pp. 1102-1117.

Delicato, F. C., Pires, P. F., Pirmez, L. and Batista, T. (2010) Wireless Sensor Networks as a Service. In: *Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on.* pp. 410-417.

Digi International (2010a) *Drop-in Networking.* Available from: http://www.digi.com/technology/drop-in-networking/ (Accessed 01/09/2010).

Digi International (2010b) *X-CTU Software - Drivers - Digi International.* Available from: http://www.digi.com/support/productdetail?pid=3352 (Accessed 16/12/2012).

Digi International (2010c) *ZigBee®/Mesh End-toEnd Wireless Modules, Adapters, Gateways.* Available from: http://www.digi.com/products/wireless/ZigBee-mesh/ (Accessed 31/08/2010).

Dong-Hyuk, C., Jung Il, L., Dong-Sung, K. and Woo Chool, P. (2006) Design and Implementation of Wireless Fieldbus for Networked Control Systems. In: *SICE-ICASE, 2006. International Joint Conference.* pp. 1036-1040.

Doshi, K. (2009) *ENTERPRISE SERVICE BUS.* Available from: http://www.mphasis.com (Accessed 04/06/2010).

Electronic Design (2009) *Reduce The Small-Memory-Footprint Requirements In Wireless Sensor Networks.* [Online Article] Available from: http://electronicdesign.com/article/communications/page/3/reduce-the-small-memory-footprint-requirements-in-.aspx. (Accessed 13/01/2010).

Ember (2012) *Ember ZigBee Wireless Networking Systems | Silicon Labs.* Available from: http://www.silabs.com/products/wireless/ZigBee/Pages/default.aspx/pdf/103007_ember_znetfact.pdf (Accessed 6/11/2012).

EPSG (2003) *Ethernet PowerLink Communication Profile Specification V. 2.0.* [Standard] Available from: http://www.ethernet-powerlink.org (Accessed 15/12/2009).

Erl, T. (2009) *SOA design patterns.* Prentice Hall PTR.

Feng, C., Talanis, T., German, R. and Dressler, F. (2009) Real-time enabled IEEE 802.15.4 sensor networks in industrial automation. In: *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on.* pp. 136-139.

Feng-Li, L., Moyne, J. R. and Tilbury, D. M. (2001) Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet. *Control Systems Magazine, IEEE,* 21 (1), pp. 66-83.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California.

Flammini, A., Marioli, D., Sisinni, E. and Taroni, A. (2009) Design and Implementation of a Wireless Fieldbus for Plastic Machineries. *Industrial Electronics, IEEE Transactions on,* 56 (3), pp. 747-755.

FTDI (2012) *FTDI Chip Home Page.* Available from: http://www.ftdichip.com/ (Accessed 15/10/2012).

Fuse Team (2010) *Open Source OSGi ESB - FUSE ESB 4 (ServiceMix 4).* Available from: http://fusesource.com/products/enterprise-servicemix/ (Accessed 03/09/2010).

Garcs-Erice, L., Bauer, D. and Scotton, P. (2009) A flexible and scalable message broker for sensor network integration. In: *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE.* Dublin, Ireland: ACM,

Gilart-Iglesias, V., Macia, P., x, rez, F., Marcos-Jorquera, D., Mora-Jimeno, F. J., Gil, M. and nez-Abarca, J. A. (2011) Integration of Business and Manufacturing Processes through Industrial Machinery as a Service Approach. In: *Services Computing (SCC), 2011 IEEE International Conference on.* pp. 751-752.

Gill, D. D. (2002). *Precision Replication of Co-Molded Meso and Micro Optics Through Injection Molding*. PhD. North Carolina State University.

Glombitza, N., Lipphardt, M., Werner, C. and Fischer, S. (2009a) Using graphical process modeling for realizing SOA programming paradigms in sensor networks. In: *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on.* pp. 61-70.

Glombitza, N., Pfisterer, D. and Fischer, S. (2009b) Integrating wireless sensor networks into web service-based business processes. In: *Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks.* Urbana Champaign, Illinois: ACM,

Gnad, A., Kratzig, M., Rauchhaupt, L. and Trikaliotis, S. (2008) Relevant influences in wireless automation. In: *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on.* pp. 341-348.

Google Inc. (2011) *Gadgets API — Google Developers.* Available from: https://developers.google.com/gadgets/ (Accessed 1/06/2011).

Greener, J. and Wimberger-Friedl, R. (2006) *Precision injection molding: process, materials, and applications.* Hanser Verlag.

Guinard, D. and Trifa, V. (2009a) Towards the Web of Things: Web Mashups for Embedded Devices. In: *WWW2009* Madrid Spain, p. 1.

Guinard, D. and Trifa, V. (2009b) Towards the Web of Things: Web Mashups for Embedded Devices. In: *WWW2009.* Madrid Spain, p. 1.

Gungor, V. C., Akan, O. B. and Akyildiz, I. F. (2008) A Real-Time and Reliable Transport (RT)2 Protocol for Wireless Sensor and Actor Networks. *Networking, IEEE/ACM Transactions on,* 16 (2), pp. 359-370.

Gungor, V. C. and Hancke, G. P. (2009) Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *Industrial Electronics, IEEE Transactions on,* 56 (10), pp. 4258-4265.

Guo, W., Healy, W. M. and Zhou, M. (2012) Impacts of 2.4-GHz ISM band interference on IEEE 802.15. 4 wireless sensor network reliability in buildings. *Instrumentation and Measurement, IEEE Transactions on,* 61 (9), pp. 2533-2544.

Gusmeroli, S., Piccione, S. and Rotondi, D. (2012) IoT@Work automation middleware system design and architecture. In: *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on.* pp. 1-8.

Hackmann, G., Gill, C. and Roman, G.-C. (2007) Extending BPEL for Interoperable Pervasive Computing. In: *Pervasive Services, IEEE International Conference on.* pp. 204-213.

Haehniche, J. and Rauchhaupt, L. (2000) Radio communication in automation systems: the R-fieldbus approach. In: *Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on.* pp. 319-326.

Halder, S. and Kim, W. (2011) Adaptive Filtering for Indoor Localization using ZigBee RSSI and LQI measurement. *IEEE Transactions on Systems, Man, and Cybernetics*.

Hanssmann, M., Sokwoo, R. and Sheng, L. (2009) No wiring constraints. *Industry Applications Magazine, IEEE,* 15 (4), pp. 60-65.

Hanxing, C. and Jun, T. (2009) Research on the Controller Area Network. In: *Networking and Digital Society, 2009. ICNDS '09. International Conference on.* Vol. 2, pp. 251-254.

HART Foundation (2009) *HART Field Communication Protocol Specification.* Available from: http://www.hartcomm.org/protocol/about/aboutprotocol_specs.html (Accessed 15/12/2009).

Heurtefeux, K. and Valois, F. (2012) Is RSSI a Good Choice for Localization in Wireless Sensor Network? In: *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on.* pp. 732-739.

Hongping, C., Jinhong, L. and Qizhi, S. (2010) Campus information portal based on Portal technology. In: *Artificial Intelligence and Education (ICAIE), 2010 International Conference on.* pp. 564-568.

Hossain, L., Patrick, J. D. and Rashid, M. A. (2002) *Enterprise Resource Planning Global Opportunities & Challanges.* Idea Group Publishing.

Huang, D., Liao, J., He, S. and Yuan, X. (2010) Constructing of marine information management platform based on ESB. In: *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on.* Vol. 1, pp. 266-269.

Idoudi, M., Elkhorchani, H. and Grayaa, K. (2013) Performance evaluation of Wireless Sensor Networks based on ZigBee technology in smart home. In: *Electrical Engineering and Software Applications (ICEESA), 2013 International Conference on.* pp. 1-4.

IEEE Std 802.11e (2005) IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)*, pp. 0_1-189.

IEEE Std 802.15.1 (2005) IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pp. 0_1-580.

IEEE Std 802.15.3 (2003) IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.3-2003*, pp. 0_1-315.

IEEE Std 802.15.4 (2006) IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 0_1-305.

ISA (2009) *ISA100, Wireless Systems for Automation.* Available from: http://www.isa.org/isa100 (Accessed 11/11/2009).

Islam, A. and Hansen, N. (2009) Two-component injection molding: present and future perspectives. Available from: http://www.4spepro.org/pdf/000049/000049.pdf (Accessed 05/11/2012)

Ivanov, S., Nett, E. and Schemmer, S. (2008) Automatic WLAN localization for industrial automation. In: *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on.* pp. 93-96.

Jackbe Corporation (2011) *Enterprise Mashup Developer Community.* Available from: http://mdc.jackbe.com/enterprise-mashup/

Jayasumana, S. (2012) *Achieving Optimal ESB Performance: Comparing Message Transfer Mechanisms using WSO2 ESB.* Available from: http://wso2.com/library/articles/2012/03/wso2-esb-message-transfer-mechanisms-comparative-benchmarks

Jecht, U., Stripf, W. and P, W. (2004) *Profibus—Open solutions for the world of automation.* 1 ed. (The Industrial Information Technology Handbook) CRC Press.

Jennic (2010) *Jennic Technology for a Changing World.* Available from: http://www.jennic.com/ (Accessed 27/08/2010).

Jennic Ltd (2010a) *DR1048 Sensor Board - JN-RM-2030-DR1048-Sensor-Board-1v4.pdf.* Available from: http://www.jennic.com/files/support_files/JN-RM-2030-DR1048-Sensor-Board-1v4.pdf

Jennic Ltd (2010b) *Product Brief – JN5148-EK010 ZigBee PRO Evaluation Kit.* Available from: http://www.jennic.com/files/product_briefs/JN5148-EK010_PB_v1.1.pdf (Accessed 28/08/2010).

Jennic Ltd (2011) *Packet Error Rate Testing - JN-AN-1006-PER-Test-Software.pdf.* Available from: http://www.jennic.com/files/support_documentation/JN-AN-1006-PER-Test-Software.pdf (Accessed 17/7/2011).

Jestratjew, A. and Kwiecien, A. (2013) Performance of HTTP Protocol in Networked Control Systems. *Industrial Informatics, IEEE Transactions on,* 9 (1), pp. 271-276.

Jin, Y., Benatallah, B., Casati, F. and Daniel, F. (2008) Understanding Mashup Development. *Internet Computing, IEEE,* 12 (5), pp. 44-52.

Karnouskos, S., Colombo, A. W., Bangemann, T., Manninen, K., Camp, R., Tilly, M., Stluka, P., Jammes, F., Delsing, J. and Eliasson, J. (2012) A SOA-based architecture for empowering future collaborative cloud-based industrial automation. In: *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society.* pp. 5766-5772.

Kayaalp, M., Ceylan, O., Bagci, I. E. and Tavli, B. (2009) Data processing and communication strategies for lifetime optimization in wireless sensor networks. In: *Signal Processing and Communications Applications Conference, 2009. SIU 2009. IEEE 17th.* pp. 769-771.

Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J. and Verschueren, P. (2004) *Patterns: Implementing an SOA Using an Enterprise Service Bus.* First Edition ed. Redbooks.

Kjellsson, J., Vallestad, A. E., Steigmann, R. and Dzung, D. (2009) Integration of a Wireless I/O Interface for PROFIBUS and PROFINET for Factory Automation. *Industrial Electronics, IEEE Transactions on,* 56 (10), pp. 4279-4287.

Kotsopoulos, K. (2010). *Design, Development and testing of a message-based Network Management platform for the integration of heterogeneous management systems*. PhD. University of Bradford.

Kotsopoulos, K., Hu, Y.-F. and Lei, P. (2010). *Managing Next Generation Networks (NGNs) based on the Service-Oriented Architechture (SOA). Design, Development and testing of a message-based Network Management platform for the integration of heterogeneous management systems.* School of Engineering Design and Technology.

Koubaa, A., Severino, R., Alves, M. and Tovar, E. (2009) Improving Quality-of-Service in Wireless Sensor Networks by Mitigating "Hidden-Node Collisions". *Industrial Informatics, IEEE Transactions on,* 5 (3), pp. 299-313.

Krasteva, Y. E., Portilla, J., Carnicer, J. M., de la Torre, E. and Riesgo, T. (2008) Remote HW-SW reconfigurable Wireless Sensor nodes. In: *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE.* pp. 2483-2488.

Kuhn, M., Cemin, Z., Merkl, B., Depeng, Y., Yazhou, W., Mahfouz, M. and Fathy, A. (2008) High accuracy UWB localization in dense indoor environments. In: *Ultra-Wideband, 2008. ICUWB 2008. IEEE International Conference on.* Vol. 2, pp. 129-132.

Kutlu, A., Ekiz, H. and Powner, E. T. (1996a) Performance analysis of MAC protocols for wireless control area network. In: *Parallel Architectures, Algorithms, and Networks, 1996. Proceedings. Second International Symposium on.* pp. 494-499.

Kutlu, A., Ekiz, H. and Powner, E. T. (1996b) Wireless control area network. In: *Networking Aspects of Radio Communication Systems , IEE Colloquium on.* pp. 3/1-3/4.

Kwei-Jay, L. and Panahi, M. (2010) A real-time service-oriented framework to support sustainable cyber-physical systems. In: *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on.* pp. 15-21.

Lantow, B. (2009) On the Path Processing under Energy Constraints. In: *Networks, 2009. ICN '09. Eighth International Conference on.* pp. 94-98.

Lavric, A., Popa, V., Finis, I., Gaitan, A. and Petrariu, A. (2012) Packet Error Rate analysis of IEEE 802.15.4 under 802.11g and Bluetooth interferences. In: *Communications (COMM), 2012 9th International Conference on.* pp. 259-262.

Lee, J. S., Chuang, C. C., & Shen, C. C. (2009, May). Applications of short-range wireless technologies to industrial automation: a ZigBee approach. In*Telecommunications, 2009. AICT'09. Fifth Advanced International Conference on* (pp. 15-20). IEEE.

Leguay, J., Lopez-Ramos, M., Jean-Marie, K. and Conan, V. (2008) An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. In: *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on.* pp. 740-747.

Leitner, S. H., & Mahnke, W. (2006). OPC UA–service-oriented architecture for industrial applications. ABB Corporate Research Center.

Li, J., Wang, L. and Yan, L. (2012) Cloud Service based intelligent power monitoring and early-warning system. In: *Innovative Smart Grid Technologies - Asia (ISGT Asia), 2012 IEEE.* pp. 1-4.

Liqun,H., & Bergmann, N. W. (2012). Novel industrial wireless sensor networks for machine condition monitoring and fault diagnosis. Instrumentation and Measurement, IEEE Transactions on, 61(10), 2787-2798.

LIU, S., ZHAO, L. J., LI, B., & ZHANG, L. B. (2015). Design and Implementation of Injection Molding Machine Equipment Monitoring System. Journal of Chongqing University of Technology (Natural Science), 5, 015.

Li, Z. (2006) ZigBee Wireless Sensor Network in Industrial Applications. In: *SICE-ICASE, 2006. International Joint Conference.* pp. 1067-1070.

Lo Bello, L. and Toscano, E. (2009) Coexistence Issues of Multiple Co-Located IEEE 802.15.4/ZigBee Networks Running on Adjacent Radio Channels in Industrial Environments. *Industrial Informatics, IEEE Transactions on,* 5 (2), pp. 157-167.

Lopez-Gomez, M., & Tejero-Calado, J. C. (2009). A lightweight and energy-efficient architecture for wireless sensor networks. Consumer Electronics, IEEE Transactions on, 55(3), 1408-1416.

Loskyll, M., Heck, I., Schlick, J. and Schwarz, M. (2012) Context-Based Orchestration for Control of Resource-Efficient Manufacturing Processes. *Future Internet,* 4 (3), pp. 737-761.

Low, K. S., Win, W. N. N. and Er, M. J. (2005) Wireless Sensor Networks for Industrial Environments. In: *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on.* Vol. 2, pp. 271-276.

Luediger, H., Guirao, M. P., Cassioli, D. and Stenzel, E. (2007) Overview of an UWB System for LDR-L/T in Industrial and Logistics Scenarios. In: *Mobile and Wireless Communications Summit, 2007. 16th IST.* pp. 1-6.

Luyang, Z., Jiaqi, L. and Ming, Y. (2006) An Integration Research on Service-oriented Architecture (SOA) for Logistics Information System. In: *Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on.* pp. 1059-1063.

Macia-Perez, F., Gilart-Iglesias, V., Ferrandiz-Colmeiro, A., Berna-Martinez, J. V. and Gea-Martinez, J. (2009) New models of agile manufacturing assisted by

semantic. In: *Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th.* pp. 336-343

Marcelloni, F., & Vecchio, M. (2008). A simple algorithm for data compression in wireless sensor networks. Communications Letters, IEEE, 12(6), 411-413.

Mahalik, N. P. (2003) *Fieldbus Technology: Industrial Network Standards for Real-Time Distributed Control.* 1 ed. Berlin: Springer.

Mazzer, Y. and Tourancheau, B. (2009) Comparisons of 6LoWPAN Implementations on Wireless Sensor Networks. In: *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on.* pp. 689-692.

Meier, U., Witte, S., Helmig, K., Hoing, M., Schnuckel, M. and Krause, H. (2007) Performance evaluation and prediction of a bluetooth based real-time sensor actuator system in harsh industrial environments. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on.* pp. 33-37.

Merrill, D. (2006) Mashups: the New Breed of Web App. Available from: http://www.ibm.com/developerworks/xml/library/xmashups. (Accessed July 2010)

Mike, P. P. and Willem-Jan, H. (2007) Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal,* 16 (3), pp. 389-415.

Min-Jeong, A., Hong-Chul, L. and Hye-Jin, J. (2007) Design of the material control system based on service oriented architecture. In: *Control, Automation and Systems, 2007. ICCAS '07. International Conference on.* pp. 978-983.

Minguez, J., Zor, S. and Reimann, P. (2011) Event-driven business process management in Engineer-to-Order supply chains. In: *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on.* pp. 624-631.

Miorandi, D. and Vitturi, S. (2005) A wireless extension of Profibus DP based on the Bluetooth radio system. *Ad Hoc Networks,* 3, pp. 479-494.

Nahapetian, A., Lombardo, P., Acquaviva, A., Benini, L. and Sarrafzadeh, M. (2007) Dynamic Reconfiguration in Sensor Networks with Regenerative Energy Sources. In: *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07.* pp. 1-6.

National Instruments (2010) *NI LabVIEW - Improving the Productivity of Engineers and Scientists - National Instruments.* Available from: http://www.ni.com/LabVIEW/

National Instruments (2011a) *LabVIEW VI Templates, Example VIs, and Tools - LabVIEW 2010 Help - National Instruments.* Available from: http://zone.ni.com/reference/en-XX/help/371361G-01/lvconcepts/lv_temp_ex_vis/

National Instruments (2011b) *NI LabVIEW Web UI Builder.* Available from: http://www.ni.com/uibuilder/

National Instruments (2012a) *Building and Accessing a LabVIEW Web Service Application (ETS, VxWorks, Windows) - LabVIEW 2012 Help - National Instruments.* Files. Available from: http://zone.ni.com/reference/en-XX/help/371361J-01/lvhowto/build_web_service/

National Instruments (2012b) *LabVIEW VISA Tutorial.* Available from: http://www.ni.com/visa/, http://www.ni.com/support/visa/vintro.pdf

National Instruments (2012c) *LabVIEW Web Services - LabVIEW 2012 Help - National Instruments.* Available from: http://zone.ni.com/reference/en-XX/help/371361J-01/lvconcepts/webservices/

National Instruments (2012d) *NI CompactRIO - Rugged, High-Performance reconfigurable control and monitoring system - National Instruments.* Available from: http://www.ni.com/compactrio/ (Accessed 10/06/2012).

National Instruments (2012e) *What is a Queue? - National Instruments.* Available from: http://digital.ni.com/public.nsf/allkb/DD7DBD9B10E3E537862565BC006CC2E4

OASIS (2007) *Web Services Business Process Execution Language Version 2.0* \DestinationEIIAIITII

ODVA (2007) *Network Infrastructure for EtherNet/IP: Introduction and Considerations* ODVA. Available from: http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00035R0_Infrastructure_Guide.pdf (Accessed 17/11/2009).

OGC (2010) *Sensor Web Enablement (SWE)* Available from: http://www.opengeospatial.org/ogc/markets-technologies/swe (Accessed 15/07/2010).

OMRON (2002) *DeviceNet Wireless Unit Operation Manual.* [Manual] Available from: http://www.ia.omron.com/product/family/1444/index_fea.html. (Accessed 12/12/2009).

OPCconnect (2012) *OPC Programmers' Connection - OLE for Process Control.* Available from: http://www.opcconnect.com/ (Accessed 5/12/2012).

OPCFoundation (2012) *Home Page - OPC Foundation.* Available from: https://opcfoundation.org/ (Accessed 5/12/2012).

Oracle Corporation (2012) *MySQL :: The world's most popular open source database.* Available from: http://www.mysql.com/ (Accessed 07/062012).

OSGi Alliance (2011) *The Open Services Gateway initiative (OSGi) Alliance.* Available from: http://www.osgi.org/Main/HomePage

Paulson, L. D. (2000) More hype than Internet bytes for online ERP. *IT Professional,* 2 (5), pp. 11-15.

Pautasso, C., Zimmermann, O. and Leymann, F. (2008) Restful web services vs. "big"' web services: making the right architectural decision. In: *Proceeding of the 17th international conference on World Wide Web.* Beijing, China: ACM,

Pei-Breivold, H., Jansen, A., Sandström, K. and Crnkovic, I. (2013) Virtualize for Architecture Sustainability in Industrial Automation. In: *: Technology.*

Perera, S. (2009) *Binary-Relay: An Efficient way to pass both XML and non-XML content through Apache Synapse | WSO2 Inc.* Available from: http://wso2.com/library/articles/binary-relay-efficient-way-pass-both-xml-non-xml-content-through-apache-synapse

Petersen, S., Carlsen, S. and Talevski, A. (2008) Industrial IT revolution through wireless sensor network technologies. In: *IT Revolutions, 2008 First Conference on.* pp. 1-6.

Petrova, M., Riihijarvi, J., Mahonen, P. and Labella, S. (2006) Performance study of IEEE 802.15.4 using measurements and simulations. In: *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE.* Vol. 1, pp. 487-492.

Pingshun, D. (2011) Design and implementation of ESB based on SOA in power system. In: *Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on.* pp. 519-522.

Pinto, A. R., Montez, C., Araújo, G., Vasques, F. and Portugal, P. (2014) An approach to implement data fusion techniques in wireless sensor networks using genetic machine learning algorithms. *Information Fusion,* 15, pp. 90-101.

Pinus, H. (2004) Middleware: Past and present a comparison. June.

Pleinevaux, P. and Decotignie, J. D. (1988) Time critical communication networks: field buses. *Network, IEEE,* 2 (3), pp. 55-63.

Polymer IRC (2010) *Polymer IRC.* Available from: http://phyast4.leeds.ac.uk/pages/HomePage (Accessed 12/08/2010).

Pramudianto, F., Simon, J., Eisenhauer, M., Khaleel, H., Pastrone, C. and Spirito, M. (2013) Prototyping the Internet of Things for the future factory using a SOA-based middleware and reliable WSNs. In: *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on.* pp. 1-4.

Profibus & Profinet international (2005) *Profinet IO application layer service definition, application layer protocol specification.* PI. Available from: http://www.profibus.com/ (Accessed 17/11/2009).

Puttonen, J., Lobov, A. and Lastra, J. L. M. (2008) An application of BPEL for service orchestration in an industrial environment. In: *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on.* pp. 530-537.

Rauchhaupt, L. (2002) System and device architecture of a radio based fieldbus-the RFieldbus system. In: *Factory Communication Systems, 2002. 4th IEEE International Workshop on.* pp. 185-192.

Rettinger, A., Lösch, U., Tresp, V., d'Amato, C. and Fanizzi, N. (2012) Mining the semantic web. *Data Mining and Knowledge Discovery,* 24 (3), pp. 613-662.

Rein, S., Lehmann, S., & Guhmann, C. (2009, March). Wavelet image two-line coder for wireless sensor node with extremely little ram. In Data Compression Conference, 2009. DCC'09. (pp. 252-261). IEEE.

Richardson, L. and Ruby, S. (2007) *RESTful Web Services.* O'Reilly Media.

Roberts, D. A. (1993) `OLCHFA' a distributed time-critical fieldbus. In: *Safety Critical Distributed Systems, IEE Colloquium on.* pp. 6/1-6/3.

Ryan, K. L. K. (2009) A computer scientist's introductory guide to business process management (BPM). *Crossroads,* 15 (4), pp. 11-18.

Safari, S., Shabani, F. and Simon, D. (2014) Multirate multisensor data fusion for linear systems using Kalman filters and a neural network. *Aerospace Science and Technology*.

Salles, N., Krommenacker, N. and Lecuire, V. (2008) Performance study of IEEE 802.15.4 for industrial maintenance applications. In: *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on.* pp. 1-6.

Samaras, I. K., Gialelis, J. V. and Hassapis, G. D. (2009) Integrating Wireless Sensor Networks into Enterprise Information Systems by Using Web Services. In: *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on.* pp. 580-587.

Samaras, I. K., Gialelis, J. V. and Hassapis, G. D. (2010) A service oriented-based system for real time industrial applications. In: *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on.* pp. 1-8.

Savio, D. and Karnouskos, S. (2008) Web-service enabled wireless sensors in SOA environments. In: *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on.* pp. 952-958.

Schramm, W., Kostinger, H., Bayrhammer, K., Fiedler, M. and Grechenig, T. (2012) Developing a hospital information system ecosystem for creating new clinical collaboration methodologies. In: *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on.* pp. 101-103.

Seno, L., Vitturi, S. and Zunino, C. (2009) Analysis of Ethernet Powerlink Wireless Extensions Based on the IEEE 802.11 WLAN. *Industrial Informatics, IEEE Transactions on,* 5 (2), pp. 86-98.

Silcher, S., Minguez, J., Scheibler, T. and Mitschang, B. (2010) A service-based approach for next-generation Product Lifecycle Management. In: *Information Reuse and Integration (IRI), 2010 IEEE International Conference on.* pp. 219-224.

Simek, M., Fuchs, M., Mraz, L., Moravek, P. and Botta, M. (2011) Measurement of LowPAN Network Coexistence with Home Microwave Appliances in Laboratory and Home Environments. In: *Broadband and Wireless Computing,*

Communication and Applications (BWCCA), 2011 International Conference on. pp. 292-299.

Singh, A. (2012) Agent Based Framework for Semantic Web Content Mining. *International Journal of Advancements in Technology,* 3 (2), pp. 108-113.

Sleman, A., Alafandi, M. and Moeller, R. (2009) Integration of wireless Fieldbus and wired Fieldbus for health monitoring. In: *Consumer Electronics, 2009. ICCE '09. Digest of Technical Papers International Conference on.* pp. 1-2.

Sleman, A. and Moeller, R. (2008) Integration of Wireless Sensor Network Services into other Home and Industrial networks; using Device Profile for Web Services (DPWS). In: *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on.* pp. 1-5.

Snatkin, A., Karjust, K., Majak, J., Aruväli, T., & Eiskop, T. (2013). Real time production monitoring system in SME. Estonian Journal of Engineering, 19(1), 62-75

Spiess, P., Karnouskos, S., Guinard, D., Savio, D., Baecker, O., de Souza, L. M. S. a. and Trifa, V. (2009a) SOA-Based Integration of the Internet of Things in Enterprise Services. In: *{IEEE} International Conference on Web Services, {ICWS 2009} , Los Angeles, CA, USA*. pp. 968--975.

Spiess, P., Karnouskos, S., Souza, L., Savio, D., Guinard, D., Trifa, V., Baecker, O. and Koehler, M. (2009b) Reliable execution of business processes on dynamic networks of service-enabled devices. In: *Proc. 7th {IEEE} International Conference on Industrial Informatics {INDIN} 2009, Cardiff, UK.* pp. 533--538.

Srinivasan, K. (2006) and Philip Levis,“RSSI is Under Appreciated”. In: *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets 2006).*

Staniec, K. (2012) Evaluation of the ZigBee Transmission Repetition Mechanism in the Variably-Loaded Reverberation Chamber.

Starke, G., Kunkel, T. and Hahn, D. (2013) Flexible collaboration and control of heterogeneous mechatronic devices and systems by means of an event-driven, SOA-based automation concept. In: *Industrial Technology (ICIT), 2013 IEEE International Conference on.* pp. 1982-1987.

Stojcev, M. K., Kosanovic, M. R. and Golubovic, L. R. (2009) Power management and energy harvesting techniques for wireless sensor nodes. In: *Telecommunication in Modern Satellite, Cable, and Broadcasting Services, 2009. TELSIKS '09. 9th International Conference on.* pp. 65-72.

Subaashini, K., Dhivya, G. and Pitchiah, R. (2013) ZigBee RF signal strength for indoor location sensing - Experiments and results. In: *Advanced Communication Technology (ICACT), 2013 15th International Conference on.* pp. 50-57.

Suk, L., Kyung Chang, L., Man Hyung, L. and Harashima, F. (2002) Integration of mobile vehicles for automated material handling using Profibus and IEEE 802.11 networks. *Industrial Electronics, IEEE Transactions on,* 49 (3), pp. 693-701.

Sun Labs (2010) *SUN SPOT World - program the world.* Available from: http://www.sunspotworld.com/index.html (Accessed 31/08/2010).

Tahir, H. and Javed, M. Y. (2009) Service guarantees in wireless sensor networks. In: *Emerging Technologies, 2009. ICET 2009. International Conference on.* pp. 433-436.

Takruri, M., Challa, S. and Yunis, R. (2009) Data fusion techniques for auto calibration in wireless sensor networks. In: *Information Fusion, 2009. FUSION '09. 12th International Conference on.* pp. 132-139.

Tanaka, S., Fujita, N., Yanagisawa, Y., Terada, T. and Tsukamoto, M. (2008) Reconfigurable hardware architecture for saving power consumption on a sensor node. In: *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on.* pp. 405-410.

Tang, S. H., Tan, Y. J., Sapuan, S. M., Sulaiman, S., Ismail, N. and Samin, R. (2007) The use of Taguchi method in the design of plastic injection mould for reducing warpage. *Journal of Materials Processing Technology,* 182 (1-3), pp. 418-426.

Tang, Z., Zeng, P., Yu, H. and Wang, H. (2008) ZigBee-based wireless extension of FOUNDATION fieldbus. In: *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on.* pp. 661-666.

Tao, G., Zheng, H., Haifeng, L., Feng, L., Dian, Z. and Hui, T. (2012) A context-aware computing mediated dynamic service composition and reconfiguration for ubiquitous environment. In: *Internet of Things (IOT), 2012 3rd International Conference on the.* pp. 16-23.

Teranishi, T. (2010) *Tera Term Open Source Project.* Available from: http://ttssh2.sourceforge.jp/ (Accessed 10/09/2010).

Thomesse, J. P. (2004) *The WorldFIP fieldbus.* 1 ed. (The Industrial Information Technology Handbook) CRC Press.

Tu Quach, N., Jonghyun, L., Kyung Jun, G., Karpjoo, J. and Sang Boem, L. (2010) An ESB Based Micro-scale Urban Air Quality Monitoring System. In: *Networking, Architecture and Storage (NAS), 2010 IEEE Fifth International Conference on.* pp. 288-293.

U.S. Department of Energy, O. o. E. E. a. R. E. (2002) *Industrial Wireless Technology for the 21st Century.* Available from: http://www.energetics.com/resourcecenter/products/roadmaps/Documents/wireless.pdf (Accessed 09/11/2009).

Urso, M. F. (2012) *moltosenso s.r.l. - Welcome into the enhanced real world of moltosenso.* Available from: http://www.moltosenso.com/ (Accessed 16/12/2012).

van der Aalst, W. M. P., Arthur, H. M. T. H. and Mathias, W. (2003) Business process management: A survey. Springer-Verlag.

W3C (2004) Web Services Architecture. Available from: http://www.w3.org/TR/ws-arch/ (Accessed 03/08/2010)

W3C (2007a) *SOAP Version 1.2.* Available from: http://www.w3.org/TR/soap12-part1/ (Accessed 03/08/2010).

W3C (2007b) *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* Available from: http://www.w3.org/TR/wsdl20 (Accessed 03/08/2010).

Wang, L., Yu, W. and Hock, J. (2014) Multi-agent system with information fusion for intelligent lighting control.

Werb, J., Newman, M., Berry, V., Lamb, S., Sexton, D. and Lapinski, M. (2005) Improved quality of service in IEEE 802.15. 4 mesh networks. In: *Proceedings of International Workshop on Wireless and Industrial Automation.* pp. 1-6.

Westerdale, S., Lowell, L. and Kazmer, D. (2008) The Effects of Temperature and Relative Humidity on Injection Molded Part Quality. In: *SPE ANTEC.*

Whiteside, B. (2004) *Micro-moulding machines overview.* Available from: http://www.ukmig.com/page.php?id=4 (Accessed 24/08/2010).

Whiteside, B. R., Martyn, M.T., Coates, P.D., (2006) Introduction to micromolding, Precision injection molding, Hanser, pp. 250-251. In: Hanser, pp. 250-251.

Wilder, J. L., Uzelac, V., Milenkovic, A. and Jovanov, E. (2008) Runtime Hardware Reconfiguration in Wireless Sensor Networks. In: *System Theory, 2008. SSST 2008. 40th Southeastern Symposium on.* pp. 154-158.

Willig, A. (2008) Recent and Emerging Topics in Wireless Industrial Communications: A Selection. *Industrial Informatics, IEEE Transactions on,* 4 (2), pp. 102-124.

Willig, A., Matheus, K. and Wolisz, A. (2005) Wireless Technology in Industrial Networks. *Proceedings of the IEEE,* 93 (6), pp. 1130-1151.

WSO2 Inc (2011a) *Enterprise Service Bus Documentation - Enterprise Service Bus 4.6.0.* Available from: http://docs.wso2.org/display/ESB460/Enterprise+Service+Bus+Documentation

WSO2 Inc (2011b) *WSO2 Carbon Platform.* Code. Available from: http://wso2.com/products/carbon/ (Accessed 1/03/2011).

WSO2 Inc (2011c) *WSO2 lean.enterprise.middleware.* Available from: http://wso2.com/

WSO2 Inc (2012a) *Google Gadget Basics - Gadget Server 1.4.0 - WSO2 Documentation.* Available from: http://docs.wso2.org/wiki/display/GS140/Google+Gadget+Basics

WSO2 Inc (2012b) *Proxy Services - Enterprise Service Bus 4.2.0 - WSO2 Documentation.* Available from: http://docs.wso2.org/wiki/display/ESB470/Proxy+Services

WSO2 Inc (2012c) *Viskit.js Charting Library by wso2.* Available from: http://wso2.github.io/viskit/

WSO2 Inc (2012d) *WSO2 Developer Studio.* Available from: http://wso2.com/products/developer-studio

WSO2 Inc (2012e) *WSO2 Business Process Server.* (SVN). Available from: http://wso2.com/products/business-process-server/

WSO2 Inc (2012f) *WSO2 Try It Tool.* Available from: http://docs.wso2.org/display/ESB451/Try+It

Xin, M. and Wei, L. (2008) The Analysis of 6LowPAN Technology. In: *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on.* Vol. 1, pp. 963-966.

Xu, N., Subbu, K. and Tang, S. (2009) Confounded factor effects on battery life in wireless sensor networks. In: *Digital Information Management, 2009. ICDIM 2009. Fourth International Conference on.* pp. 1-6.

Yao, Z., Liang, D., Jiang, W., Bo, H. and Yuzhuo, F. (2008) Implementing indoor positioning system via ZigBee devices. In: *Signals, Systems and Computers, 2008 42nd Asilomar Conference on.* pp. 1867-1871.

Yasmeen, K. A., Wahiduzzaman, A. K. M., Imtiaz, A. and Matin, M. A. (2008) Performance analysis of ultra wide band indoor channel. In: *Communications, Propagation and Electronics, 2008. MIC-CPE 2008. 2008 Mosharaka International Conference on.* pp. 1-10.

Zhan, J., Meng, W. and Lanfen, L. (2010) Research on Service-Oriented Reconfigurable Business Collaboration Platform for Cluster Supply Chain. In: *Manufacturing Automation (ICMA), 2010 International Conference on.* pp. 203-209.

Zhao, J., Chen, G. and Juay, Y. (2007) Development of process monitoring technologies for polymer micro moulding process. Citeseer.

Zhao, J., Mayes, R., Dumoulin, L. and Tay, L. (2001) Injection moulding control studies. In: *Polymer Process Engineering 01.* London: IOM Communication Ltd, pp. 18-27.

Zhihua, W., Xiaoyu, Z., Xinkai, C., Lingwei, Z. and Hanjun, J. (2008) An energy-efficient ASIC with real-time work-on-demand for wireless body sensor network. In: *Electron Devices and Solid-State Circuits, 2008. EDSSC 2008. IEEE International Conference on.* pp. 1-6.

Zhou, R., Liu, H., He, S. and Wang, D. (2009) Data Processing and Node Management in Wireless Sensor Network. In: *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on.* pp. 1-4.

Zhuang, L. Q., Goh, K. M. and Zhang, J. B. (2007) The wireless sensor networks for factory automation: Issues and challenges. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on.* pp. 141-148.

ZigBee Alliance (2006) *ZigBee-2006 Specification.* Available from: http://www.ZigBee.org/Products/DownloadZigBeeTechnicalDocuments.aspx (Accessed 15/10/2009).

ZigBee Alliance (2007) *ZigBee Pro Specification.* Available from: http://www.ZigBee.org/Products/DownloadZigBeeTechnicalDocuments.aspx (Accessed 15/10/2009).

Zunino, C. (2007) Performance measurements of 802.11 WLANs with burst background traffic. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on.* pp. 1392-1395.

Zurawski, R. (2004) *The Industrial Information Technology Handbook.* 1 ed. CRC Press.

Appendices

Appendix A. WSN Tools and Methods

A.1 Methodology for programming node firmware:

A typical WSN consists of a number of geographically distributed sensor nodes which cooperate to monitor the physical qualities of a given environment. A wireless sensor node in a network consists of four basic components: sensing unit, processing unit, transceiver unit and a power unit (Akyildiz et al., 2002). In order for the sensor node to carry out the monitoring of its environment, the components of the sensor node have to communicate with each other. The on-board or external sensors measure some physical quantity and convert it into a signal using the ADC to be processed by the processing unit. The integrated transceiver unit is then used to send the processed data through RF communication with other nodes and devices on the network. This is all done through the use of a software/firmware program which resides in the memory of the sensor nodes processing unit. There are three stages to program a typical sensor node.

Stage 1 Developing and Compiling: The firmware has to be written and then compiled using an Interactive Development Environment (IDE). The programming language used to develop the firmware program varies depending upon the manufacturer and the development tools that have been used for this purpose. Once the firmware program has been created on an IDE it is compiled into binary format using the relevant compiler.

Stage 2 Connecting to sensor node: Once we have the compiled binary file it is ready to be transferred over to the sensor node unit. This is typically done over the serial/UART port and requires the sensor node to be physically connected via a USB or serial cable. Most manufacturers also support over the air programming of the sensor node in which case the firmware is transferred over to the sensor node through a wireless connection.

Stage 3: Configuring the sensor node: The sensor node has to be placed into programming mode before the firm ware can be written. Each manufacturer has its own procedure for placing the sensor node into programming mode.

Stage 4: Configuring the Flash Utility: The firmware is downloaded onto the sensor node using either a separate flash memory programming application or an integrated utility within the IDE. Once we have the compiled firmware into a binary file and connected the sensor node to the serial/USB port, the flash utility has to be configured to read a COM port to which the sensor node is connected to on the host computer. Figure A.1 shows a Jennic flash programming application setup on COM 5 to which a sensor node is connected to.

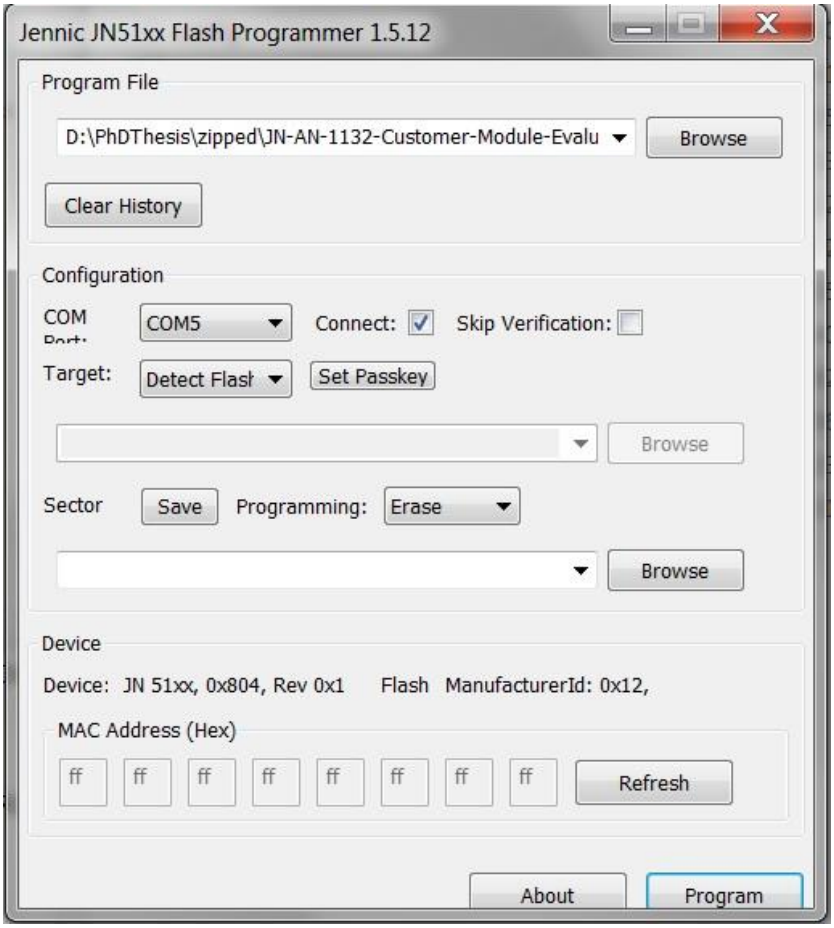


Figure A.1 Jennic Flash Utility with sensor node connected to COM port 5.

Stage 5 Uploading the Firmware: The Flash utility at this stage should be able to see the connected node with the details of the nodes setup configuration. If this is not the case then there has been an error in the previous stages which

needs to be rectified. If all is well then the node can be flashed with the binary format file by simply pressing the upload or flash button in the utility. The file is uploaded onto the sensor node and the progress can be monitored whilst the node is being flashed as shown in Figure A.2. This will typically take around one minute to complete. Once this is completed the node can be disconnected and rebooted to restart with the new firmware.

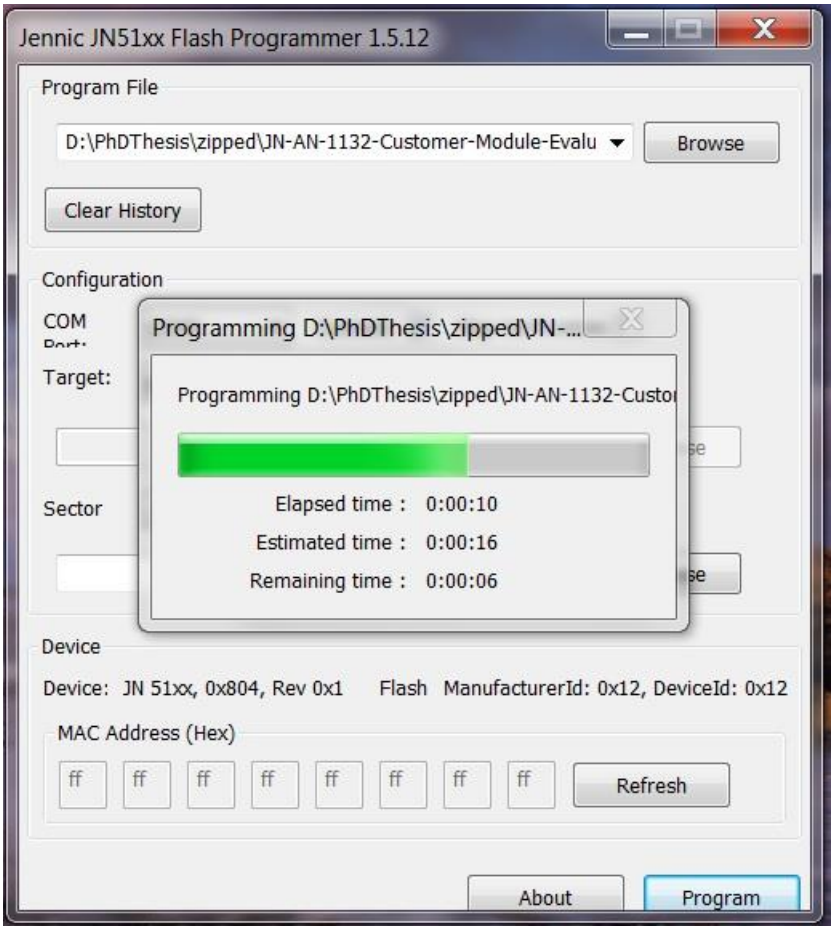


Figure A.2 Binary file being uploaded to a Jennic sensor node

Appendix B. The IM/μIM Process and Equipment

B.1 The IM/μIM Process

In this project the main business process which will be modelled and linked with other business processes in the enterprise will be the Injection Moulding process. This process is at the heart of all the manufacturing machinery in the Polymer MNT Lab at Bradford University. The Injection moulding process is a versatile technology that has been widely used to process and reprocess different type of polymers. This process has been used in almost one-third of all manufactured plastic parts (Tang et al., 2007). This process is one of the preferred processes in manufacturing industry because it can produce complex-shape plastic parts with good dimensional accuracy and very short cycle times.

B.1.1 Process Overview

For a better understanding of the core injection moulding process, the main phases in the moulding process (metering, injection, holding, and cooling) and the units in the injection moulding machines must be understood. **Error! Not a valid bookmark self-reference.** shows the main units of a typical injection moulding machine which are the clamping unit, the plasticating unit, and the drive unit. The clamping unit holds the injection mould. It is capable of closing, clamping, and opening the mould. Its main components are the fixed and moving plates, the tie bars, and the mechanism for opening, closing and clamping. The injection unit or plasticating unit melts the plastic and injects it into the mould. The drive unit provides power for the plasticating

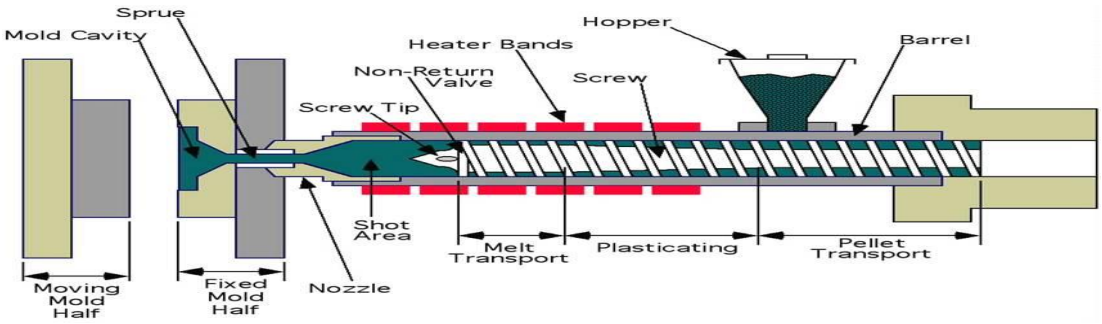


Figure B.1 Typical Injection Moulding Machine (Gill, 2002)

unit and clamping unit. An overview of the complete injection moulding cycle is given in Figure B.2. From this diagram we can see that there are four main stages in the cycle; heating and melting, followed by injection and filling, packing and holding, and cooling and ejection of the injection moulded part. On the completion of the cycle the mould closes again and the cycle starts over again.

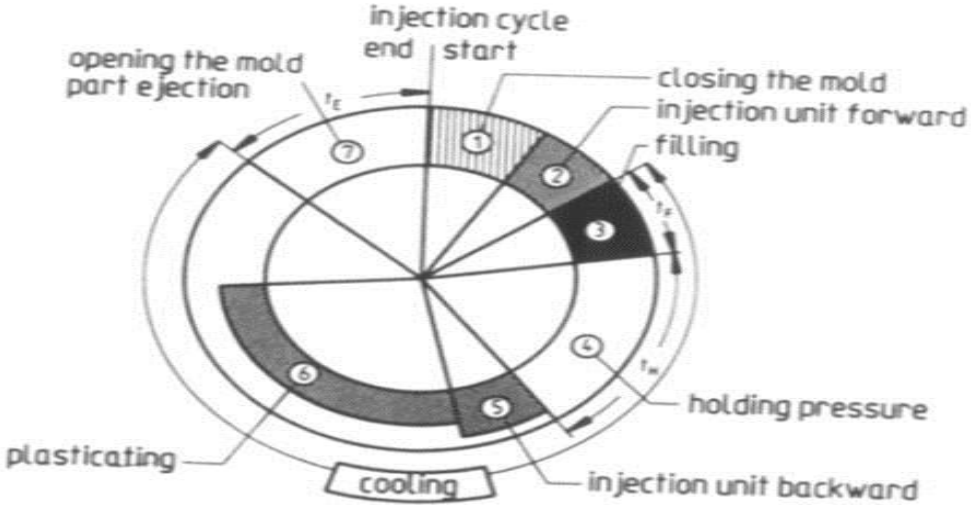


Figure B.2 injection Moulding Life Cycle

Heating and Melting Stage: In the heating stage shown in Figure B.3, plastic pellets are gravity fed through the throat of the hopper into the barrel. The electrical heater band inside the barrel as well as the shearing action of the rotating screw causes the material to melt. The most important temperature is the actual melt temperature, the temperature the materials must be heated to before it is injected unto the cavity. All materials have a temperature range at which they can be efficiently processed whilst still maintaining their optimum mechanical properties.

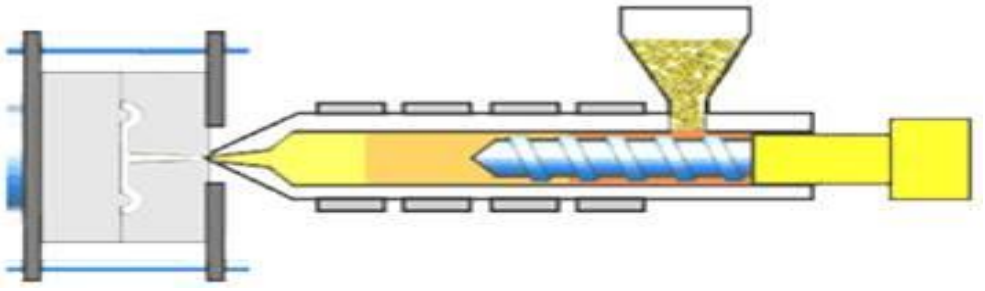


Figure B.3 Feeding, Heating and Melting Stage

Injection and Filling Stage: The mould is closed and the nozzle of the extruder is pushed against the Sprue bushing of the mould. The screw is not rotating at this point and is pushed forward so that the plastic melt in front of the screw is forced into the mould allowing the cavities to be filled as shown in Figure B.4.

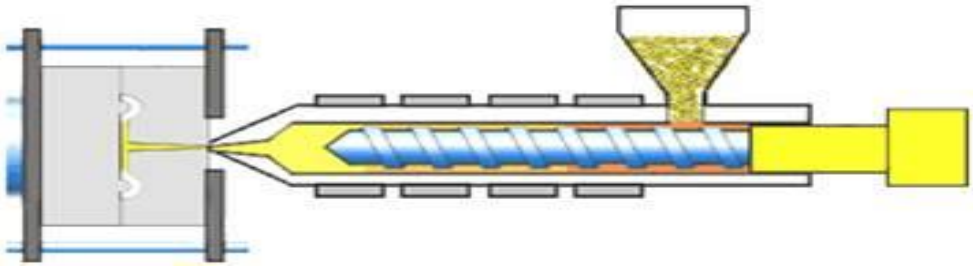


Figure B.4 Injection and Filling Stage

Packing and Holding Stage: Figure B.5 shows the packing and hold stage in which the mould is completely filled and the screw remains stationary for some time to keep the plastic in the mould under pressure; this is called the “hold” time. During the hold time additional melt is injected into the mould to compensate for contraction due to cooling and to avoid and shrinkage that may occur. When the gate freezes, the screw rotation is started. The period of screw rotation is called screw “recovery”. The rotation of the screw causes the plastic to be conveyed forward. As the plastic moves forward, heat from the electric heater bands along the barrel and shear starts to melt the plastic. At the discharge end of the screw, the plastic will be completely melted. The melt that accumulates at the end of the screw pushes the screw backward. Thus the screw is rotating and moving backward at the same time. The rate at which plastic melt accumulates in front of the screw can be controlled by the screw back pressure, that is, the hydraulic pressure exerted on the screw.

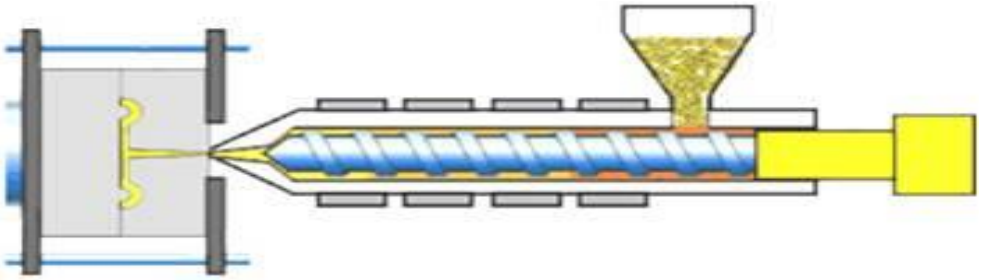


Figure B.5 Packing and Holding Stage

This also controls the melt pressure in front of the screw and also to improve thermal homogeneity significantly.

Ejection Stage: When the material in the mould has cooled sufficiently to hold its shape, the mould opens and the parts are ejected from the mould as shown in Figure B.6. When the moulded part has been ejected, the mould closes and the cycle starts over again.

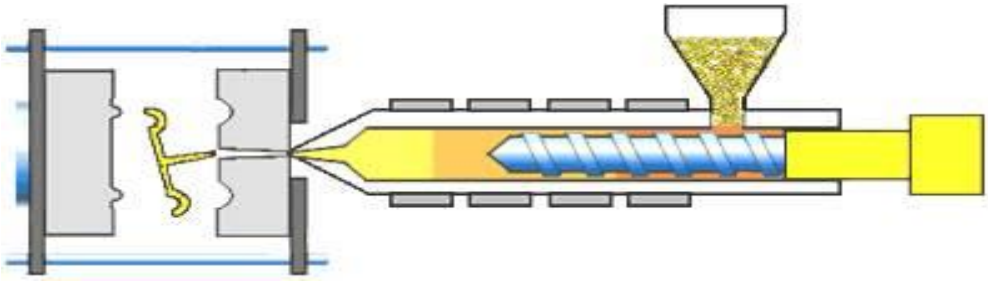


Figure B.6 Ejection Stage

B.2 The Polymer MNT Laboratory and Equipment

The Centre for Polymer Micro and Nano Technology (MNT) (Bradford University, 2010) is a facility with in the world class Polymer IRC laboratory (Polymer IRC, 2010) based at the University of Bradford. Recent advances in micro and nano-technology have realized devices with a huge range of applications in the telecommunications, medical and automotive sectors.

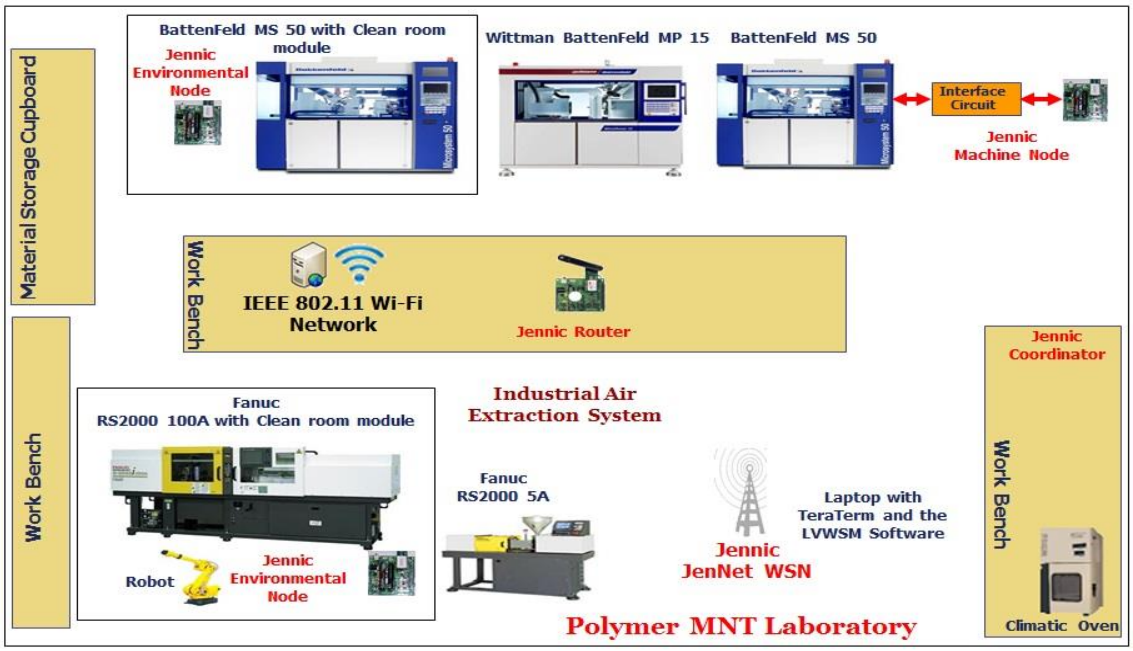


Figure B.7 Overview of the Polymer MNT Lab

The centre was setup in light of these advances and bases the micro injection moulding (micro-moulding) process at the heart of its operations. The micro injection moulding (micro-moulding) process is an optimum technique for micro-component manufacture whilst offering the benefits of conventional injection moulding process such as low marginal costs and high production capacity. The centre allows companies of all sizes to access the facilities micro injection moulding equipment and expertise in order to carry out research, experimental and development work. Figure B.7 shows the layout overview of the MNT Lab with the relevant machinery and equipment.

The Polymer MNT Lab has various processing hardware specifically designed for micro and nano moulding activities. A brief overview of this hardware is given below:

B.2.1 Battenfeld Microsystem 50 µIM Machine

The Microsystem 50 from Battenfeld is a fully electric micro-moulding machine designed specifically for the manufacturing of micro-components in the one digit

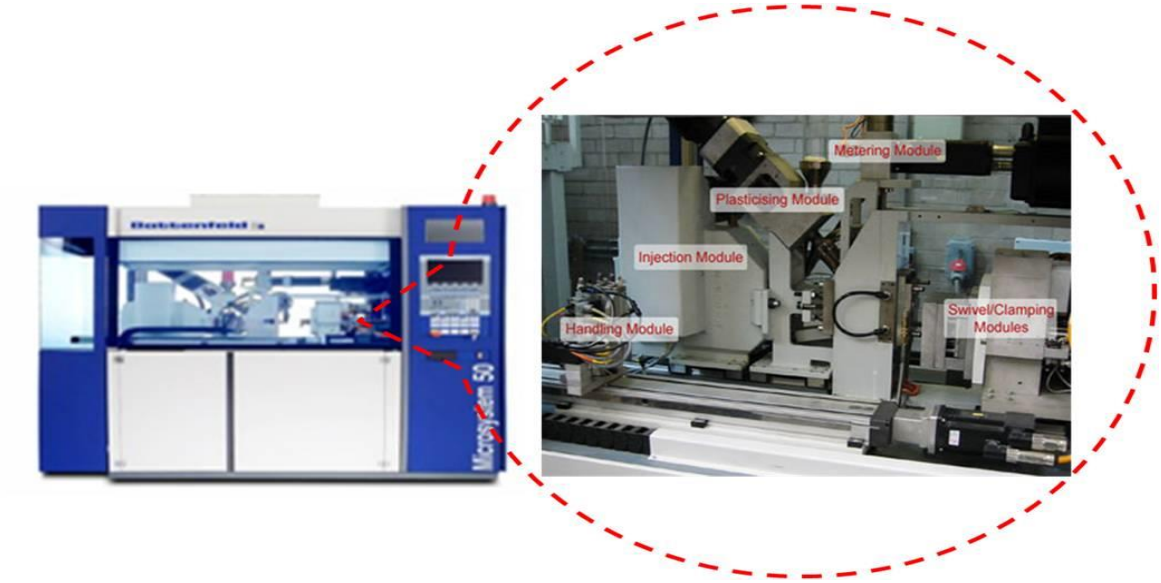


Figure B.8 The Battenfeld Microsystem 50 and its modules

milligram region. The Microsystem 50 is based on a modular concept and incorporates the ideal 3-step system which carries out plasticizing, dosing and injection as separate processes, this offers improvements over conventional injection moulding machines such as high precision, fast cycle times, low

energy consumption and less material waste. The machine consists of a number of modules which can perform moulding, removal, quality control and packing, all within a clean-room environment (Whiteside, 2004). Figure B.8 shows the Microsystem 50 and its modules.

B.2.2 Battenfeld MicroPower 15 µIM Machine

The MicroPower 15 shown in Figure B.8 is an all-electric press and is available with 5 tonnes to 15 tonnes of clamping force. It incorporates a two-step injection unit that consists of a screw and plunger and offers shot volumes ranging from 0.05 cm^3 to 4 cm^3. The innovative injection unit delivers a thermally homogeneous melt, resulting in top-quality parts, stable production processes and short cycle times. The MicroPower 15 consists of a modular basic platform that can be augmented with a parts-removal/handling system; an integrated vision-type quality-control system; and a cleanroom module that, when

Figure B.9 The Battenfeld Micropower 15 µIM machine

attached, provides the customer with a complete production cell. The benefits of the MicroPower 15 become obvious when processing expensive bio-resorbable plastic materials, such as those used in certain clips, bone screws, bone plates and other medical devices. These materials can cost between 2,000 €/kg and 5,000 €/kg, so every 1/10th of a gram of material saved underscores the equipment's cost-efficiency. Apart from the unique high quality of moulded components, other benefits of MicroPower is its cost-efficiency. Thanks to

shorter cycle times as well as lower material and energy consumption, cost savings of about 30 to 50 % can be realised compared to conventional machines.

B.2.3 Weiss WKL 100 Climatic test chamber

The Climatic test chamber by Weiss Technik shown in Figure B.10 allows the simulation of climatic conditions by allowing the setting of temperature and humidity.

Figure B.10 Weiss WKL 100 Climatic chamber

Some materials need to have very little moisture and need to be dried or be stored under certain temperature and humidity condition before they can be used in the injection moulding machines. The Weiss climatic oven is used to prepare the materials according to the requirements before being used in the injection moulding machines. For this project the climatic oven has also been used for the calibration of on board humidity and temperature sensors on the sensor nodes.

Appendix C. The Jennic Sensor node calibration data

C.1 DR1048 and DR1047 node calibration

The sensor nodes to be used in the clean rooms were placed in the WKL 100 climatic chamber. The climatic oven was set to initial values of 38 Celsius temperature and humidity to 10%. Once the chamber reached these values, the temperature was set to 18 Celsius and Humidity of 40%. As the chamber environment started to change to achieve these values, readings were taken from the both the DR1048 and DR1047 sensor nodes on-board temperature and humidity sensors. These readings were then compared with the climatic oven temperature and humidity readings. A linear curve fitting was used to derive the straight line equation in excel. This was then programmed in software to calibrate the sensors. The data for the temperature and humidity along with the climatic oven readings is shown below. The straight line graphs from excel are also shown with the equations for each sensor node.

C.1.1 On-board Temperature Sensor Calibration Data

Table C.1 shows the temperature readings from each sensor node. The temperature starts at 38C and goes down to set temperature of 18C. The temperature sensor readings from each board are noted and the corresponding climatic oven temperature readings were recorded.

Reading	Node 0x7e241	Node 0x7e39b	Climatic Oven
1	38	39	38
2	38	39	37.8
3	37	38	37
4	37	37	36.8
5	37	37	36.3
6	36	37	35.9
7	36	36	35.3
8	35	35	35
9	35	35	34.7
10	34	35	34.3
11	34	34	33.9
12	34	34	33.3

13	33	33	32.7
14	32	33	32.1
15	32	32	31.6
16	31	31	31
17	31	31	30.4
18	30	31	29.9
19	29	30	29.2
20	29	29	28.7
21	28	29	28
22	28	28	27.6
23	27	28	27.1
24	27	27	26.6
25	26	27	26.1
26	26	27	25.8
27	26	26	25.2
28	25	26	24.9
29	25	25	24.3
30	24	24	23.7
31	24	24	23.2
32	23	23	22.7
33	23	23	22.2
34	22	23	21.9
35	22	22	21.5
36	21	22	21.1
37	21	22	20.9
38	21	21	20.5
39	21	21	20.3
40	20	21	20
41	20	21	19.8
42	20	20	19.6
43	20	20	19.5
44	20	20	19.3
45	19	20	19.1
46	19	20	18.9
47	19	19	18.7
48	19	19	18.5
49	19	19	18.2

Table C.1 Temperature sensor readings: On-board temperature sensor readings from each node and the climatic oven temperature readings are shown.

C.1.1.1 *On-board Temperature Sensor Calibration Data Graphs*

The temperature starts at 38C and goes down to set temperature of 18C. The temperature sensor readings from each board are noted and the corresponding climatic oven temperature readings were recorded. A linear curve fitting was used to derive the straight line equations for each node in excel as shown in Figure C.1.

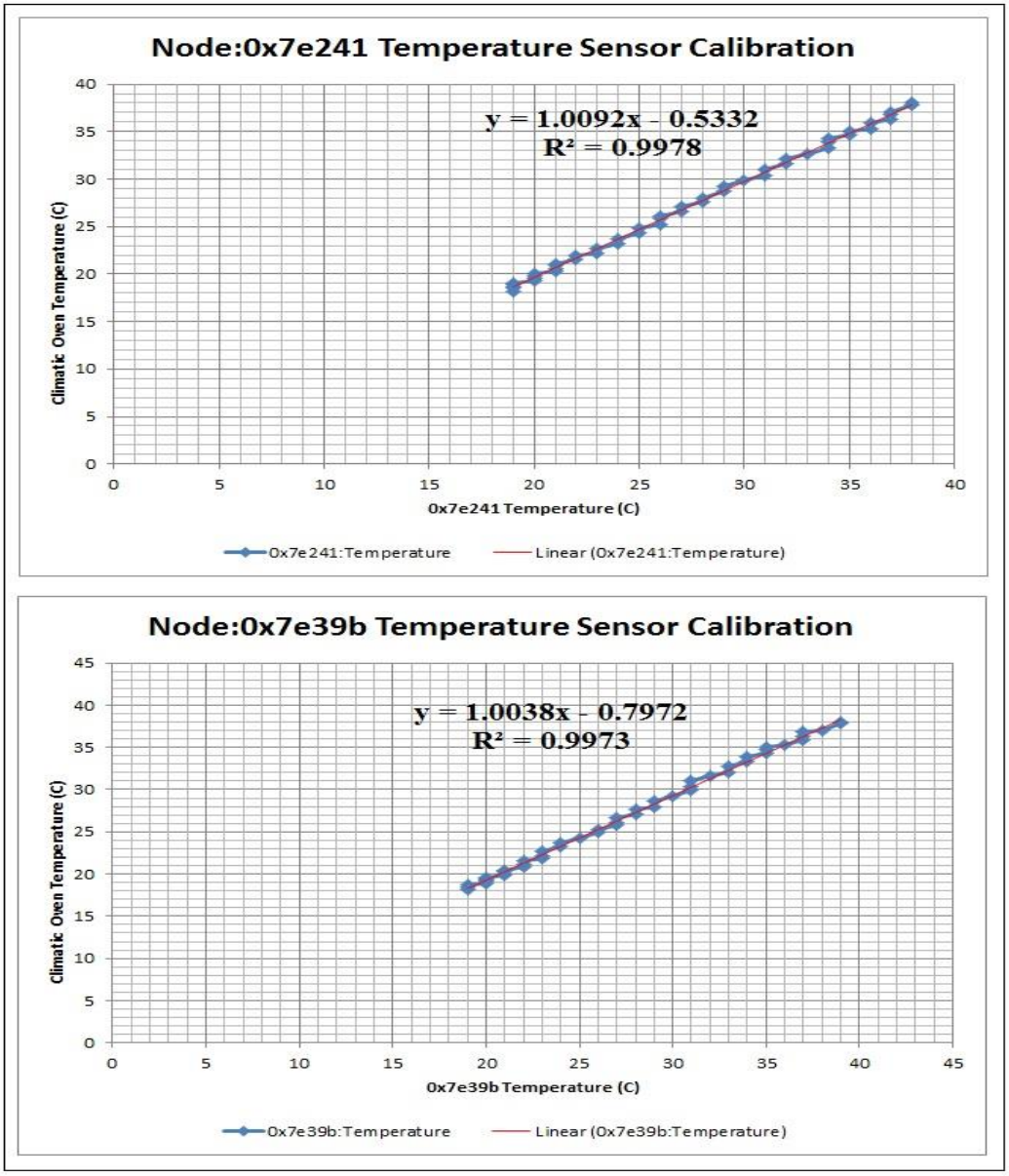


Figure C.1 Straight line equations derived for the temperature sensor on each node.

C.1.2 On-board Humidity Sensor Calibration Data

Table C.2 shows the humidity readings from each sensor node. The humidity starts at 10% and goes up to a set humidity value of 40%. The humidity sensor readings from each board are noted and the corresponding climatic oven humidity readings were recorded.

Reading	Node 0x7e24a	Node 0x7e39b	Climatic Oven	CO Rounded
1	10	12	10	10
2	10	12	10.5	11
3	10	12	11	11
4	10	12	11.1	11
5	11	13	11.2	11
6	11	14	11.5	12
7	12	14	11.8	12
8	12	14	12	12
9	12	15	12.4	12
10	13	15	13	13
11	14	15	13.1	13
12	14	16	13.6	14
13	14	16	14	14
14	15	16	14.5	15
15	15	17	14.7	15
16	15	18	15.1	15
17	16	18	15.5	16
18	16	18	16	16
19	18	19	17	17
20	18	20	17.5	18
21	19	20	18	18
22	19	21	19	19
23	19	21	20	20
24	20	20	20.5	21
25	20	21	21	21
26	21	22	21.5	22
27	21	22	22	22
28	22	23	22.6	23
29	22	23	23	23
30	23	24	23.5	24
31	23	25	24	24
32	24	25	24.6	25
33	25	26	25	25

34	26	27	25.5	26
35	26	27	26	26
36	27	28	26.7	27
37	27	28	27	27
38	28	28	27.6	28
39	28	29	28.1	28
40	29	29	28.6	29
41	29	30	29	29
42	30	31	29.5	30
43	30	31	30	30
44	31	32	30.6	31
45	31	32	31	31
46	32	32	31.5	32
47	32	33	32	32
48	33	34	32.6	33
49	33	34	33	33
50	34	35	33.7	34
51	34	35	34.2	34
52	35	37	35	35
53	35	37	35.3	35
54	36	37	35.6	36
55	36	38	35.7	36
56	36	39	35.9	36
57	37	40	36.3	36
58	37	40	36.5	37
59	37	40	36.6	37
60	38	40	36.9	37
61	38	40	37.1	37
62	38	41	37.6	38
63	39	41	38	38
64	40	42	38.5	39
65	40	42	39	39
66	41	43	39.5	40
67	41	43	39.7	40
68	41	43	39.9	40
69	41	43	40	40

Table C.2 Humidity sensor readings: On-board humidity sensor readings from each node and the climatic oven humidity readings are shown.

C.1.2.1 *On-board Humidity Sensor Calibration Data Graphs*

The Humidity starts at 10% and goes up to a set humidity of 40%. The humidity sensor readings from each board are noted and the corresponding climatic oven humidity readings were recorded. A linear curve fitting was used to derive the straight line equations for each node in excel.

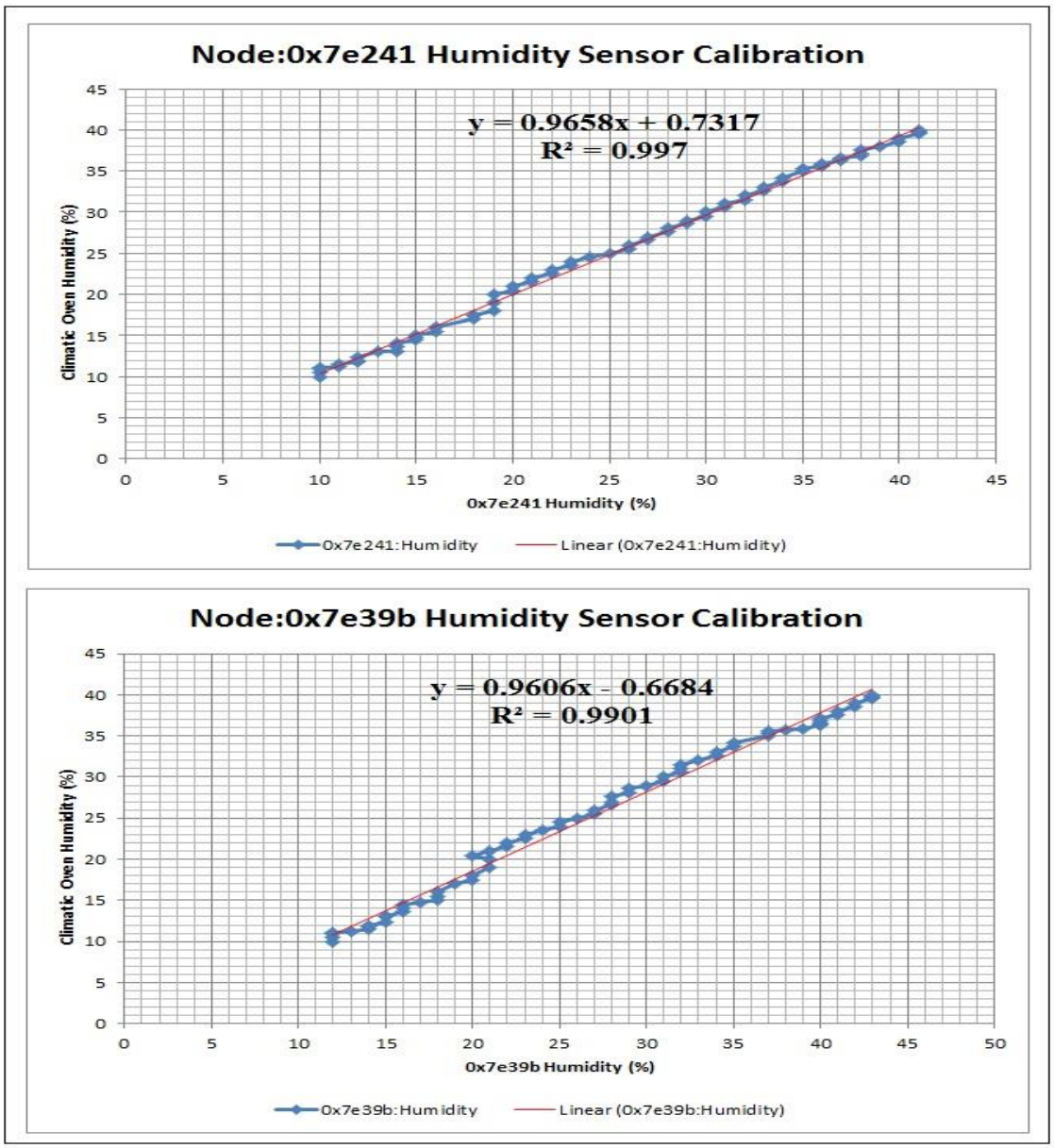
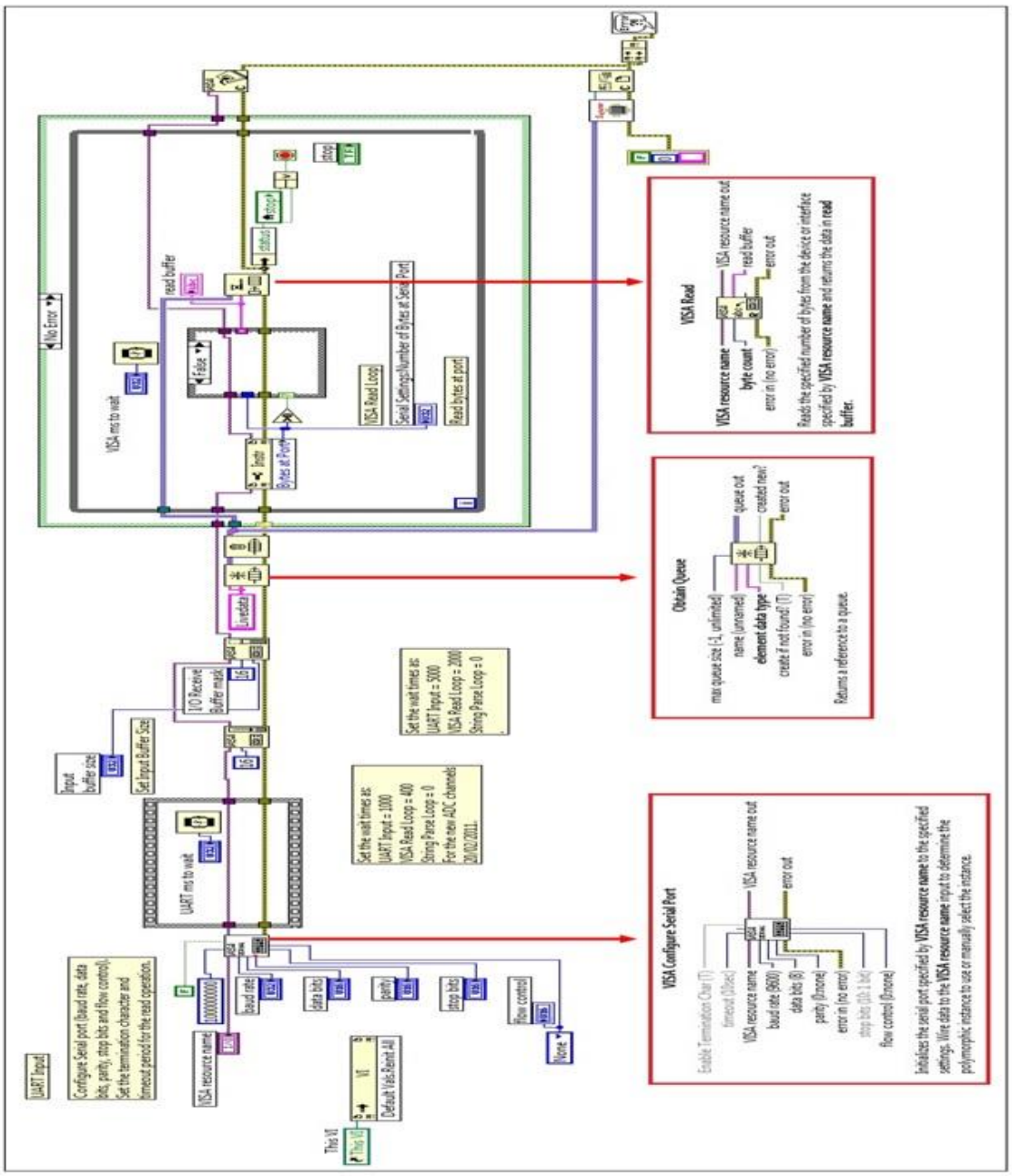


Figure C.2 Straight line equations derived for the humidity sensor on each node.

Appendix D. LVWSM Process Monitoring System Software Code

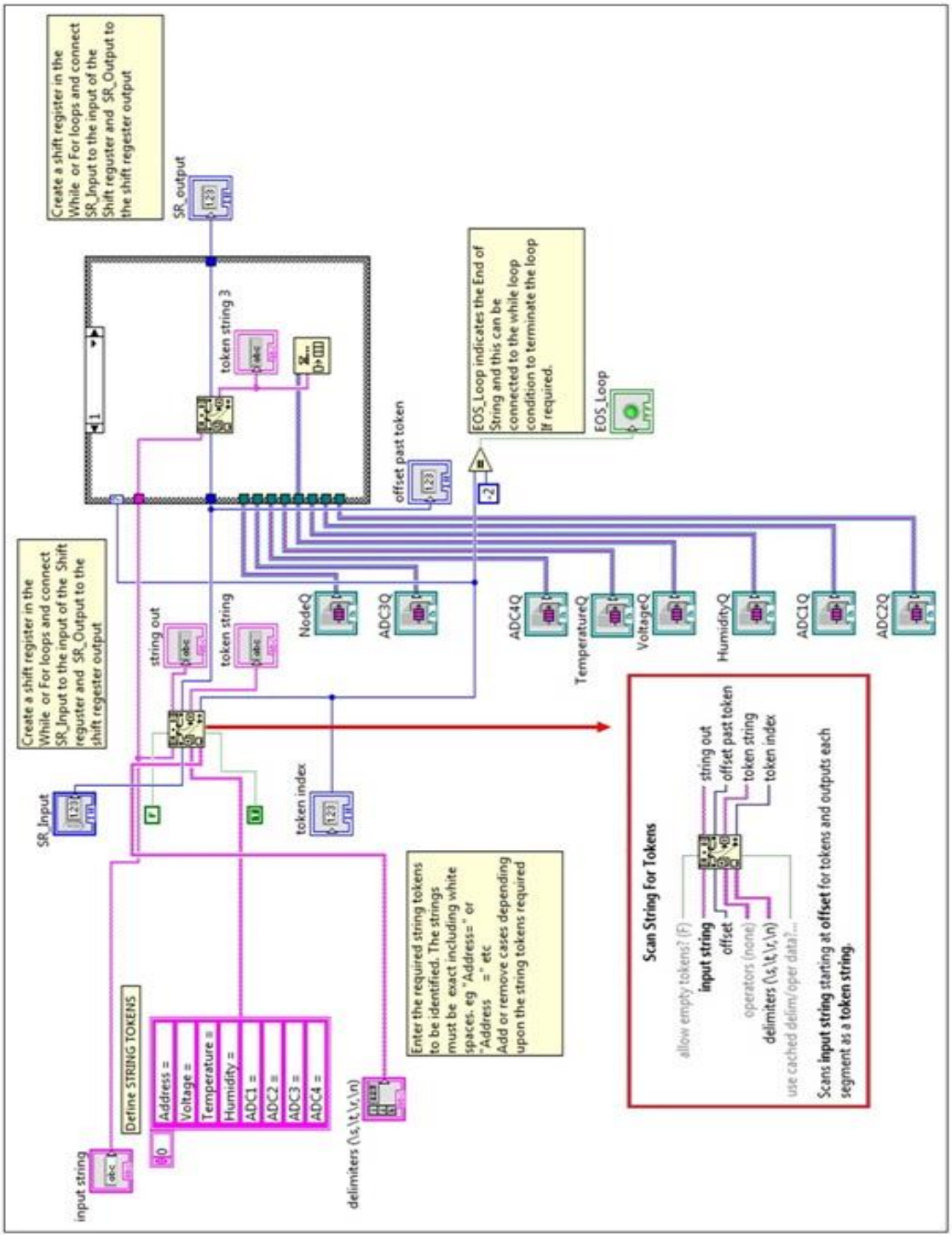
The LabVIEW Wireless Sensor Monitor (LVWSM) process monitoring system was developed in using National Instruments LabVIEW 2009 software. The following sub-sections show the complete LabVIEW code for the different components of the system.

D.1 Data Monitor Layer Code

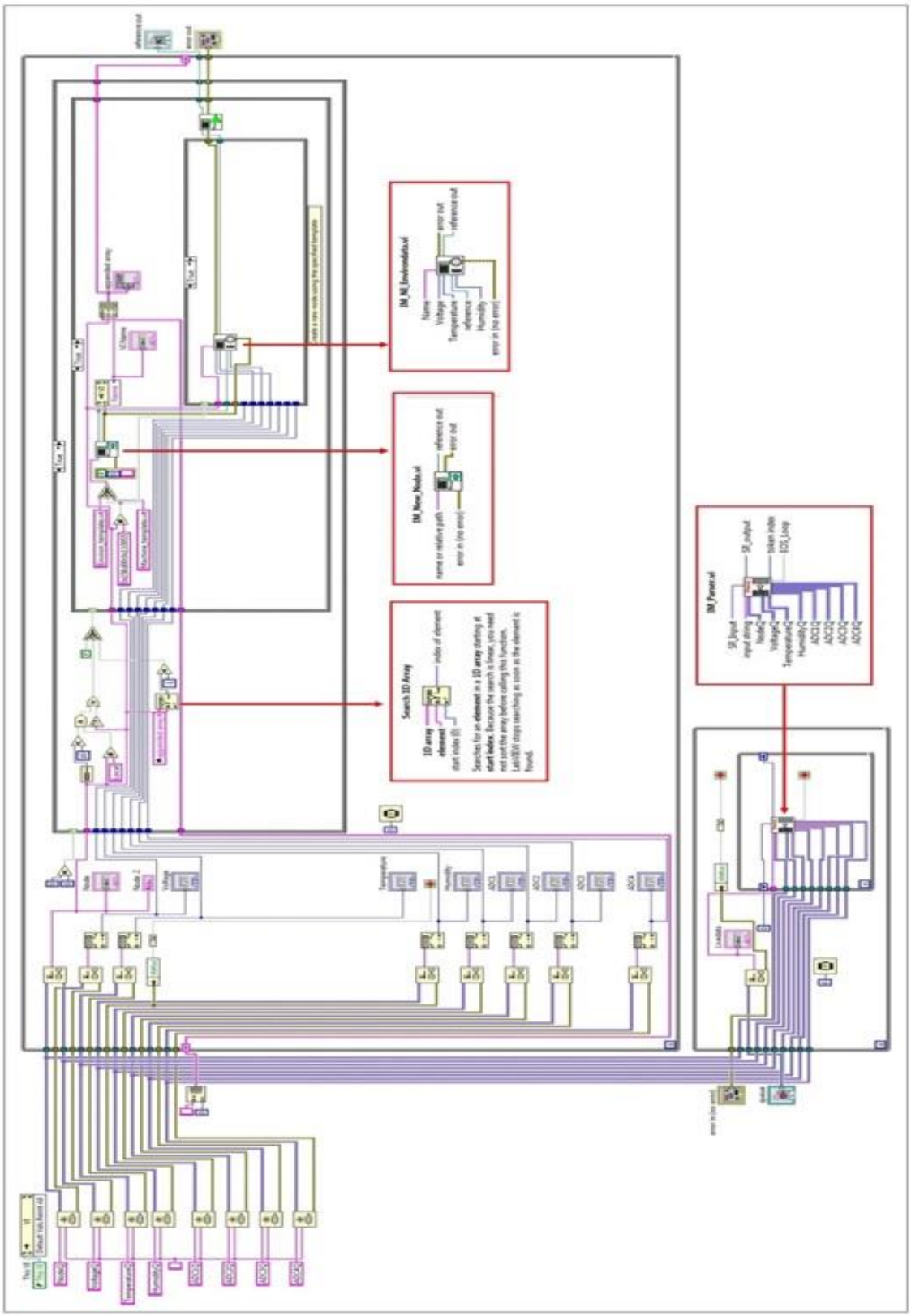


D.2 Data Processing Layer Code

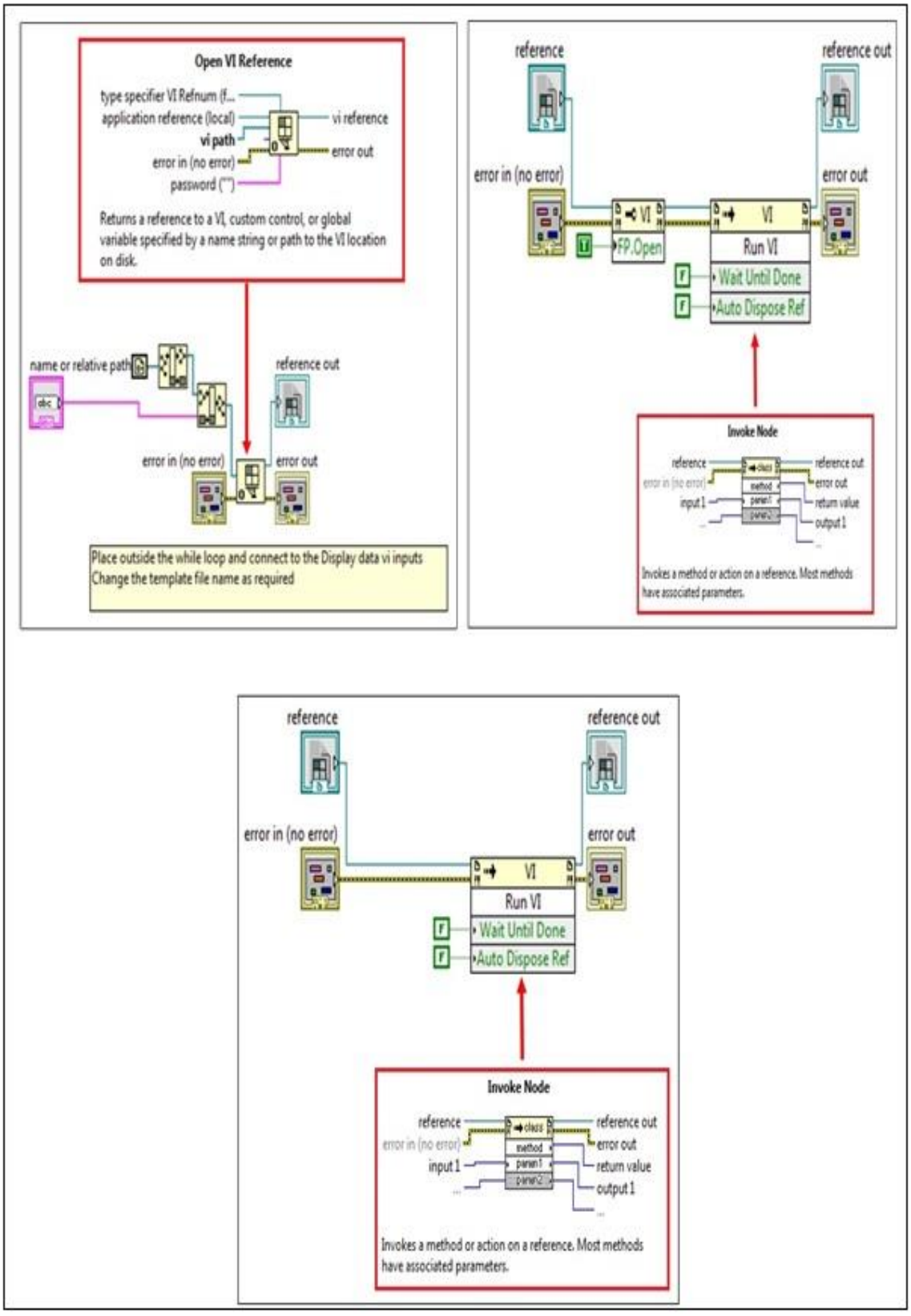
D.2.1 Data Parser Component



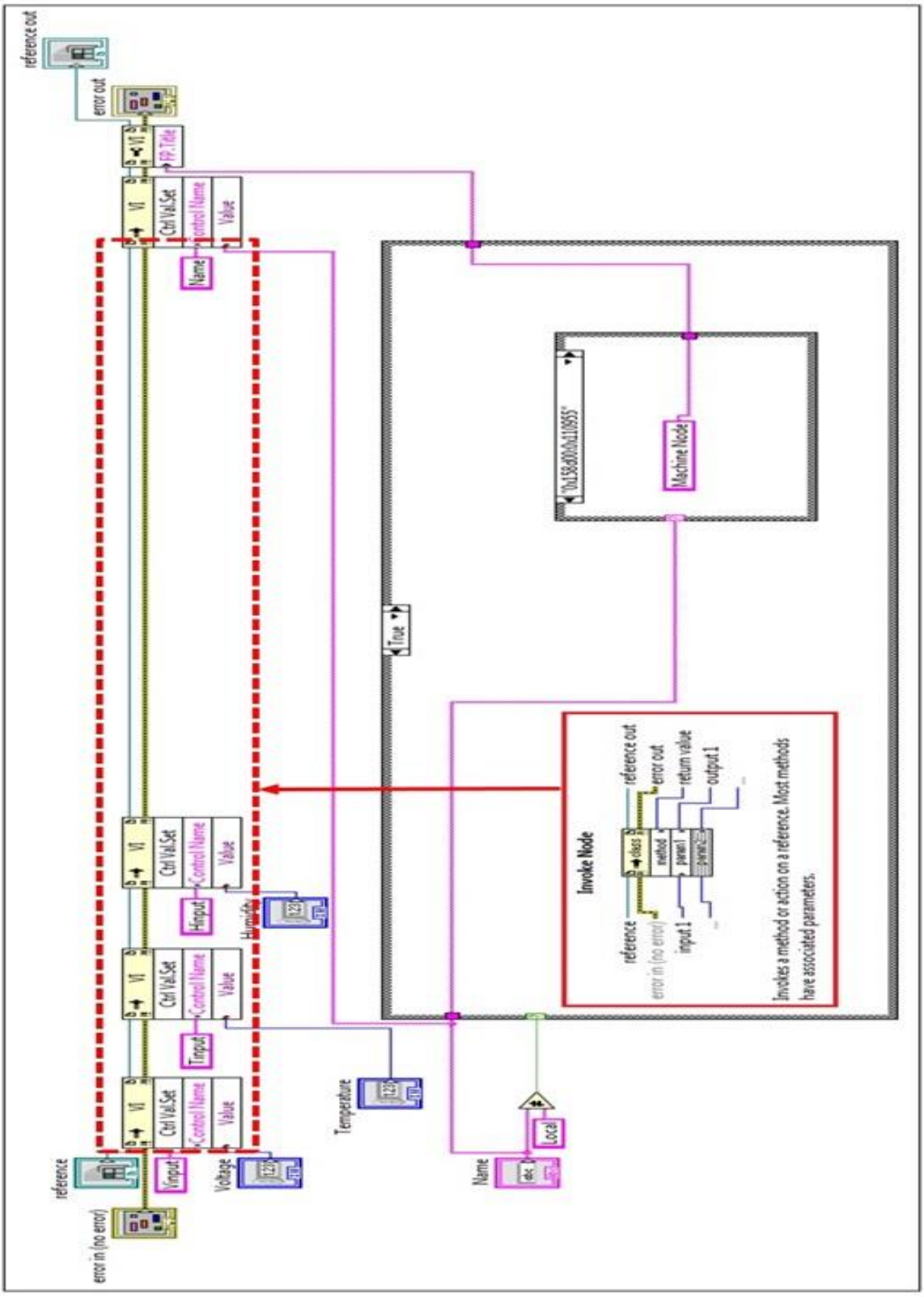
D.2.2 Data Processor Component



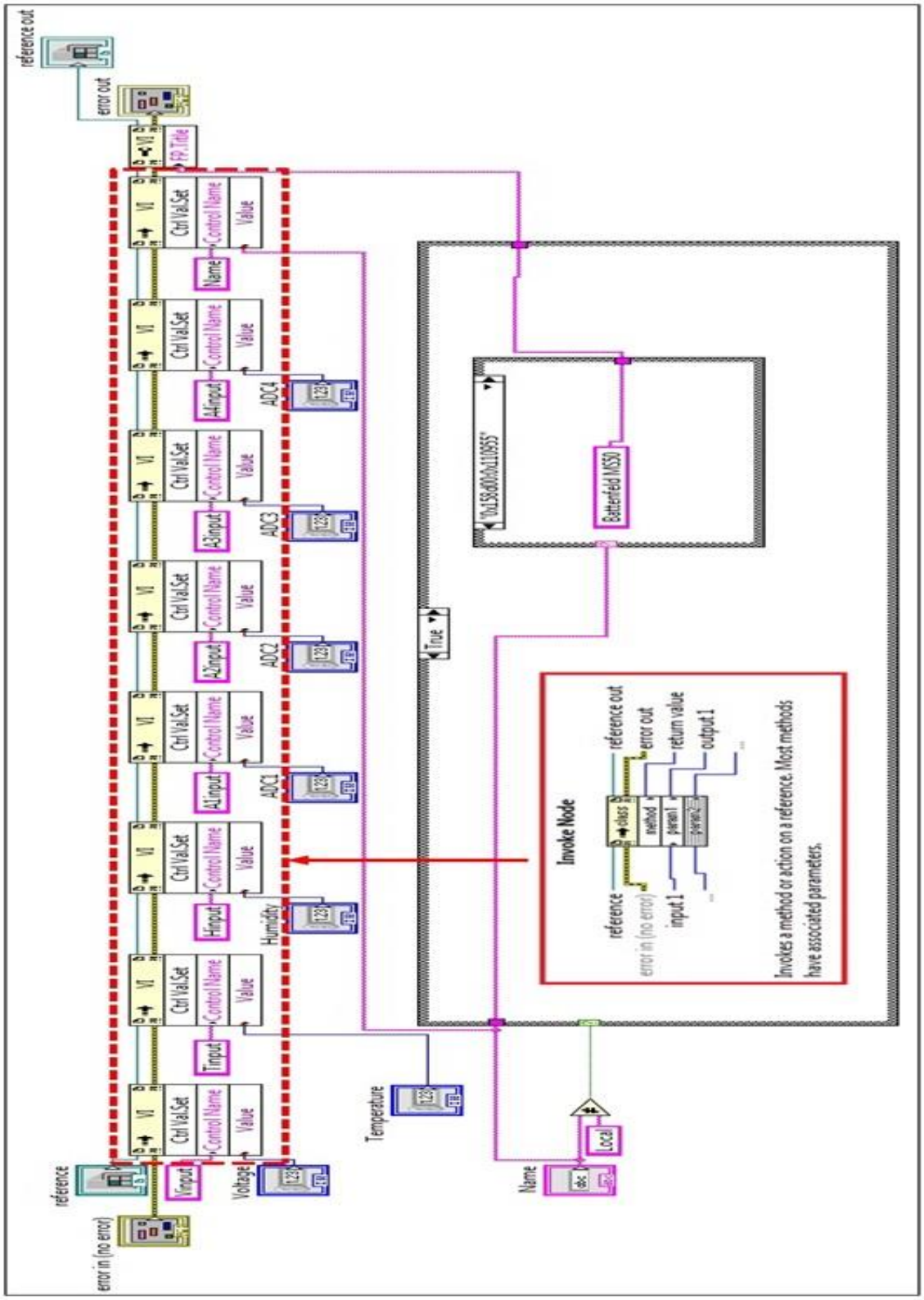
D.2.3 Node Instance Creator and Updater Components



D.2.4 Environmental Node Component

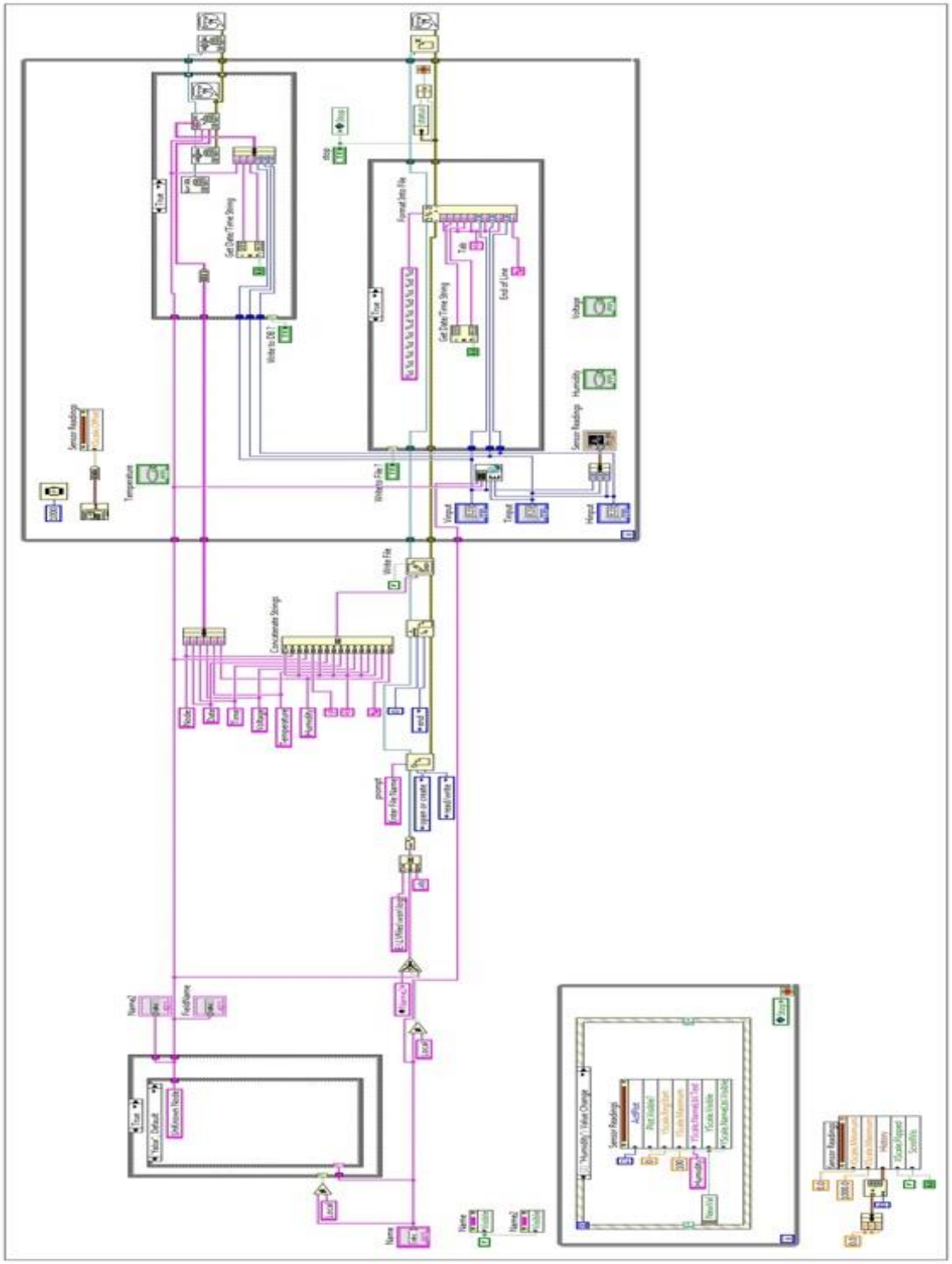


D.2.5 Machine Node Component

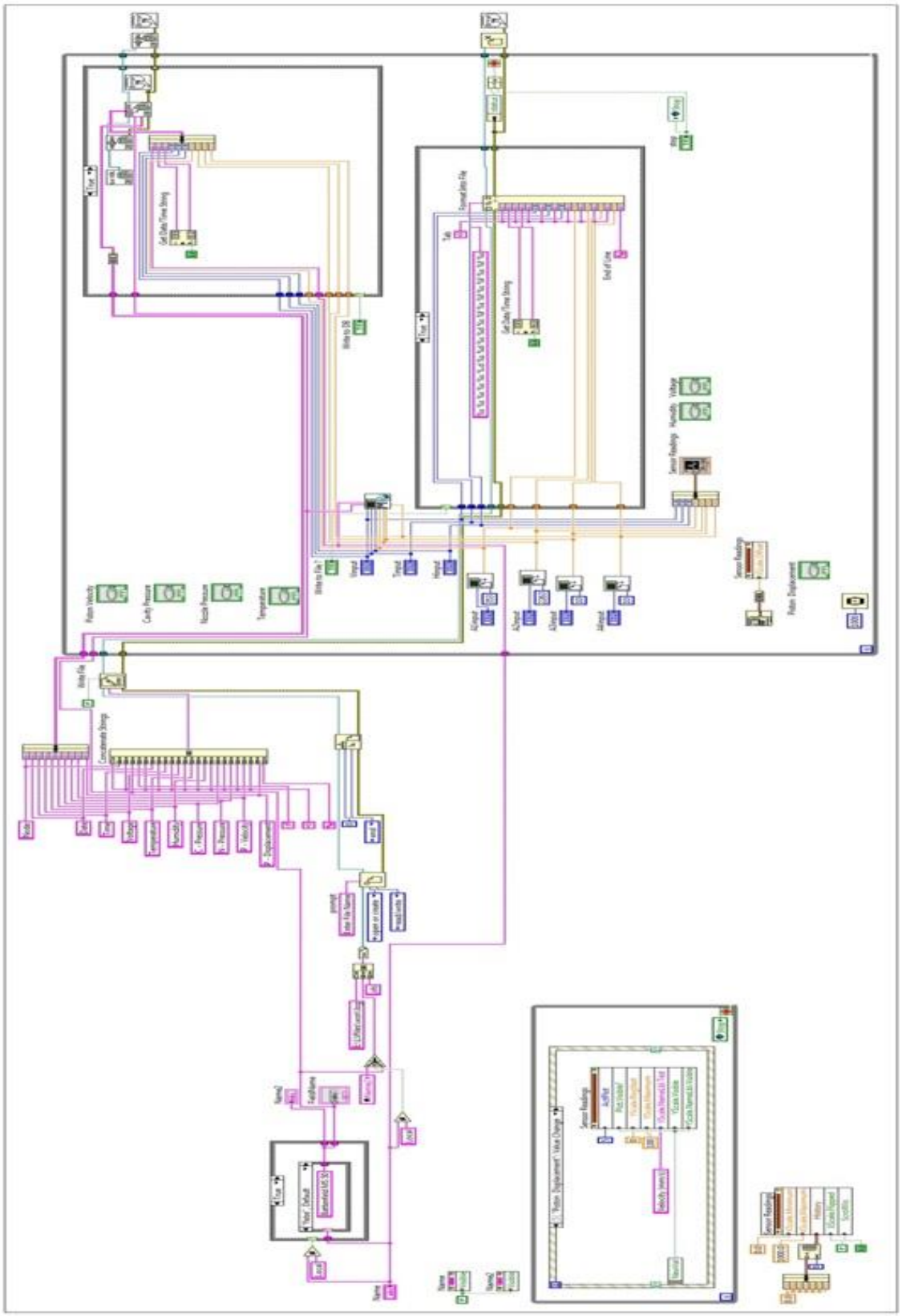


D.3 Data Storage Layer Code

D.3.1 Complete Environmental Node Template Code

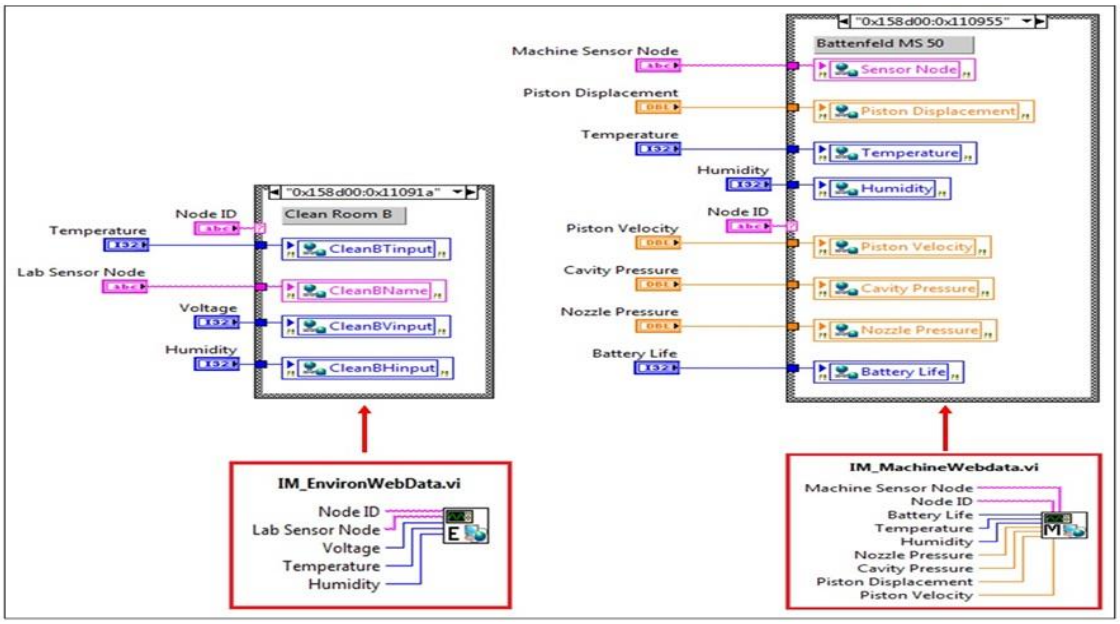


D.3.2 Complete Machine Node Template Code

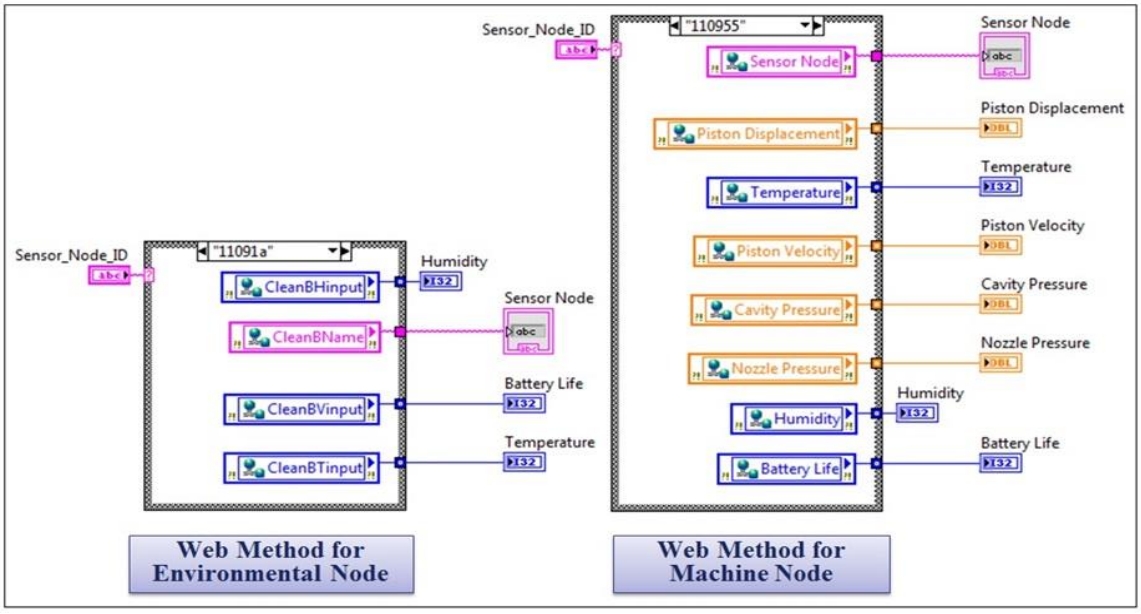


D.4 Sensor Web Layer Code

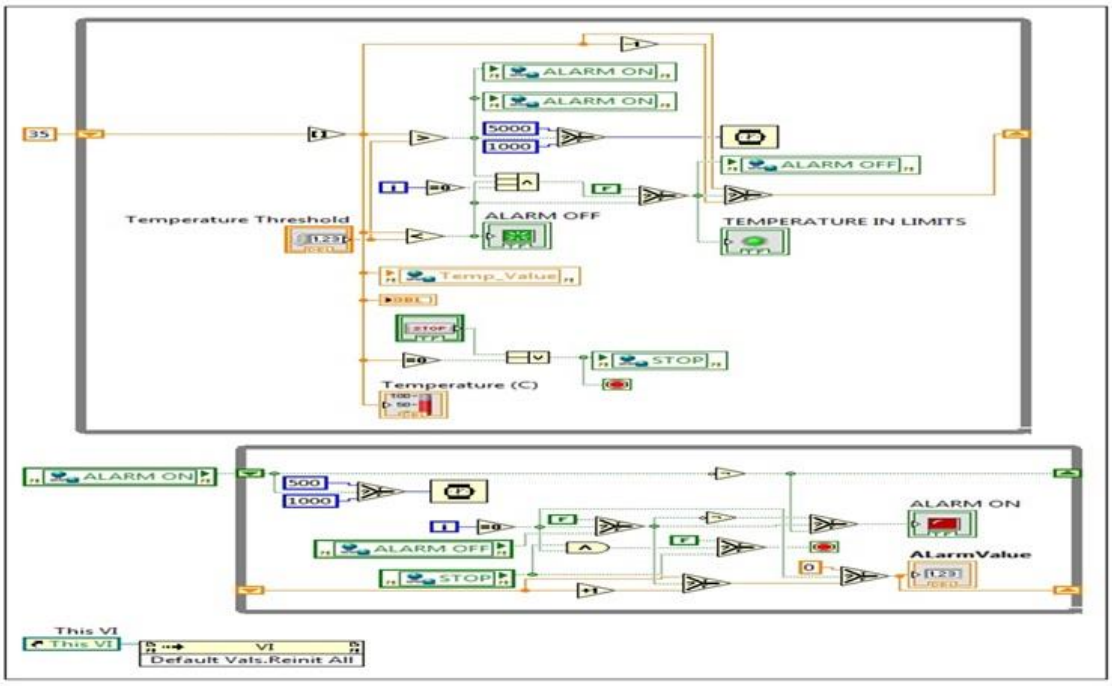
D.4.1 Environmental and Machine Web Data



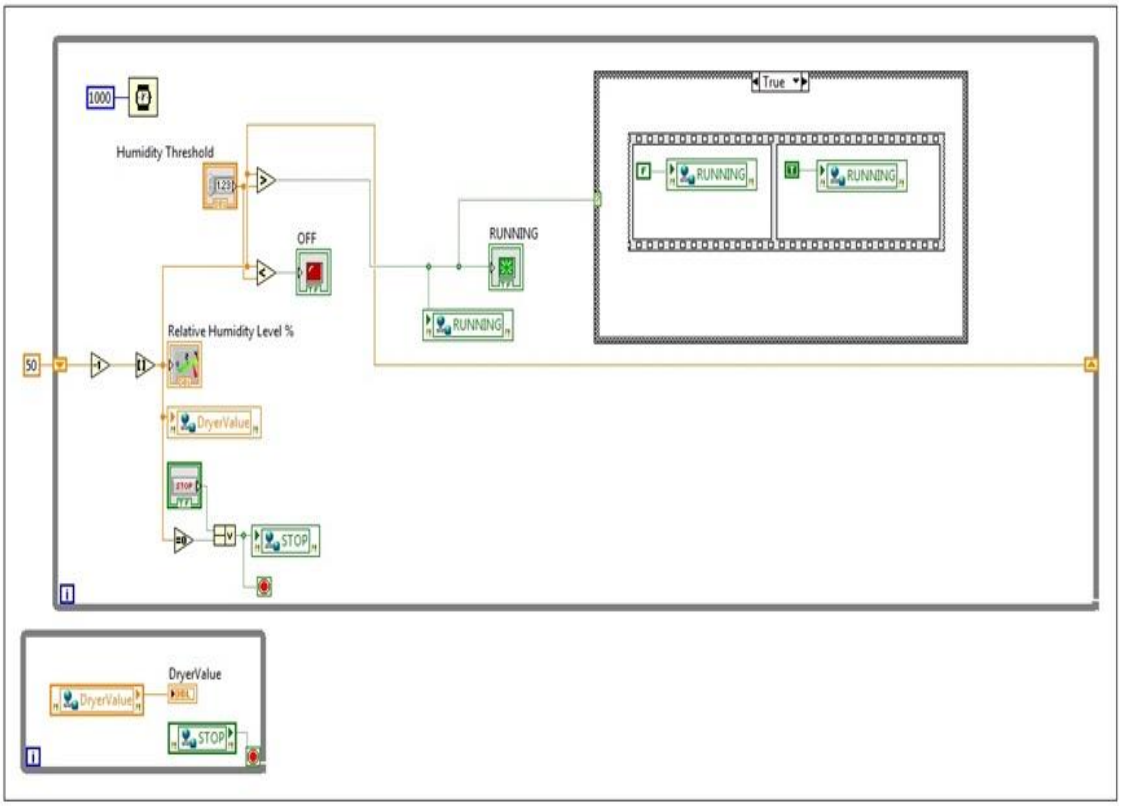
D.4.2 Environmental and Machine Web Service



D.5 Temperature Alarm Code



D.6 Material Dryer Code



Appendix E. Data Web Services XML code

E.1 Data Service XML code

```
<data name="LVDBservice" enableBoxcarring="true" serviceGroup="MySQL database">
   <description>A data service to extract data from a MYSQL database which is being updated by a Labview application
   <config id="LVDBSource">
      <property name="org.wso2.ws.dataservice.driver">com.mysql.jdbc.Driver</property>
      <property name="org.wso2.ws.dataservice.protocol">jdbc:mysql://localhost:3306/battendb</property>
      <property name="org.wso2.ws.dataservice.user">root</property>
      <property name="org.wso2.ws.dataservice.password">admin</property>
      <property name="org.wso2.ws.dataservice.xa_datasource_class"></property>
      <property name="org.wso2.ws.dataservice.xa_datasource_properties">
         <property name="URL"></property>
         <property name="User"></property>
         <property name="Password"></property>
      </property>
      <property name="org.wso2.ws.dataservice.transaction_isolation"></property>
      <property name="org.wso2.ws.dataservice.initial_size"></property>
      <property name="org.wso2.ws.dataservice.maxpoolsize"></property>
      <property name="org.wso2.ws.dataservice.minpoolsize"></property>
      <property name="org.wso2.ws.dataservice.max_wait"></property>
      <property name="org.wso2.ws.dataservice.validation_query"></property>
      <property name="org.wso2.ws.dataservice.test_on_return"></property>
      <property name="org.wso2.ws.dataservice.test_on_borrow"></property>
      <property name="org.wso2.ws.dataservice.test_while_idle"></property>
      <property name="org.wso2.ws.dataservice.time_between_eviction_runs_mills"></property>
      <property name="org.wso2.ws.dataservice.num_test_per_eviction_run"></property>
      <property name="org.wso2.ws.dataservice.min_evictable_idle_time_millis"></property>
      <property name="org.wso2.ws.dataservice.remove_abandoned"></property>
      <property name="org.wso2.ws.dataservice.remove_abandoned_timeout"></property>
      <property name="org.wso2.ws.dataservice.log_abandoned"></property>
   </config>
   <query id="LVDBAllQ" useConfig="LVDBSource">
      <sql>select Date_Time, Voltage, Temperature, Humidity from env11091a</sql>
      <result element="sensor_readings" rowName="sensor_reading">
         <element name="Date_Time" column="Date_Time" xsdType="xs:dateTime" />
         <element name="Voltage" column="Voltage" xsdType="xs:double" />
         <element name="Temperature" column="Temperature" xsdType="xs:double" />
         <element name="Humidity" column="Humidity" xsdType="xs:double" />
      </result>
      </query>
      <query id="LVDBVoltageQ" useConfig="LVDBSource">
         <sql>select Date_Time, Voltage from env11091a</sql>
         <result element="voltage_readings" rowName="VReading">
            <element name="Date_Time" column="Date_Time" xsdType="xs:dateTime" />
            <element name="Voltage" column="Voltage" xsdType="xs:double" />
         </result>
      </query>
      <query id="LVDBTempQ" useConfig="LVDBSource">
         <sql>select Date_Time, Temperature from env11091a</sql>
         <result element="temp_readings" rowName="TReading">
            <element name="Date_Time" column="Date_Time" xsdType="xs:dateTime" />
            <element name="Temperature" column="Temperature" xsdType="xs:double" />
         </result>
      </query>
      <query id="LVDBHumidityQ" useConfig="LVDBSource">
         <sql>select Date_Time, Humidity from env11091a where Date_Time in (select max(m.Date_Time) from env11091a m);</sql>
         <result element="humidity_readings" rowName="HReading">
            <element name="Date_Time" column="Date_Time" xsdType="xs:dateTime" />
            <element name="Humidity" column="Humidity" xsdType="xs:double" />
         </result>
      </query>
      <operation name="getAllReading">
         <description>Returns all sensor readings from the node with a time stamp </description>
         <call-query href="LVDBAllQ" />
      </operation>
      <operation name="getVoltageReading">
         <description>Returns the WSN node Voltage Reading from the MYSQL database&#xd; </description>
         <call-query href="LVDBVoltageQ" />
      </operation>
      <operation name="getTempReading">
         <description>Returns the WSN node Temperature Reading from the MYSQL database      &#xd;   </description>
         <call-query href="LVDBTempQ" />
      </operation>
      <operation name="getHumidityReading">
         <description>Returns the WSN node Humidity Reading from the MYSQL database  &#xd;   </description>
         <call-query href="LVDBHumidityQ" />
      </operation>
   </data>
```

E.2 WSO2 TryIt Tool

The WSO2 TryIt tool provides a quick and easy way to test Web Services WSDL file. The tool provides a mechanism of testing a WSDL by creating endpoints which use different transports on the go. The TryIt tool can be used in two ways on the WSO2 servers: Selecting it direct from the Tools side bar or "Try this Service" link that appears next to the service when it is deployed.

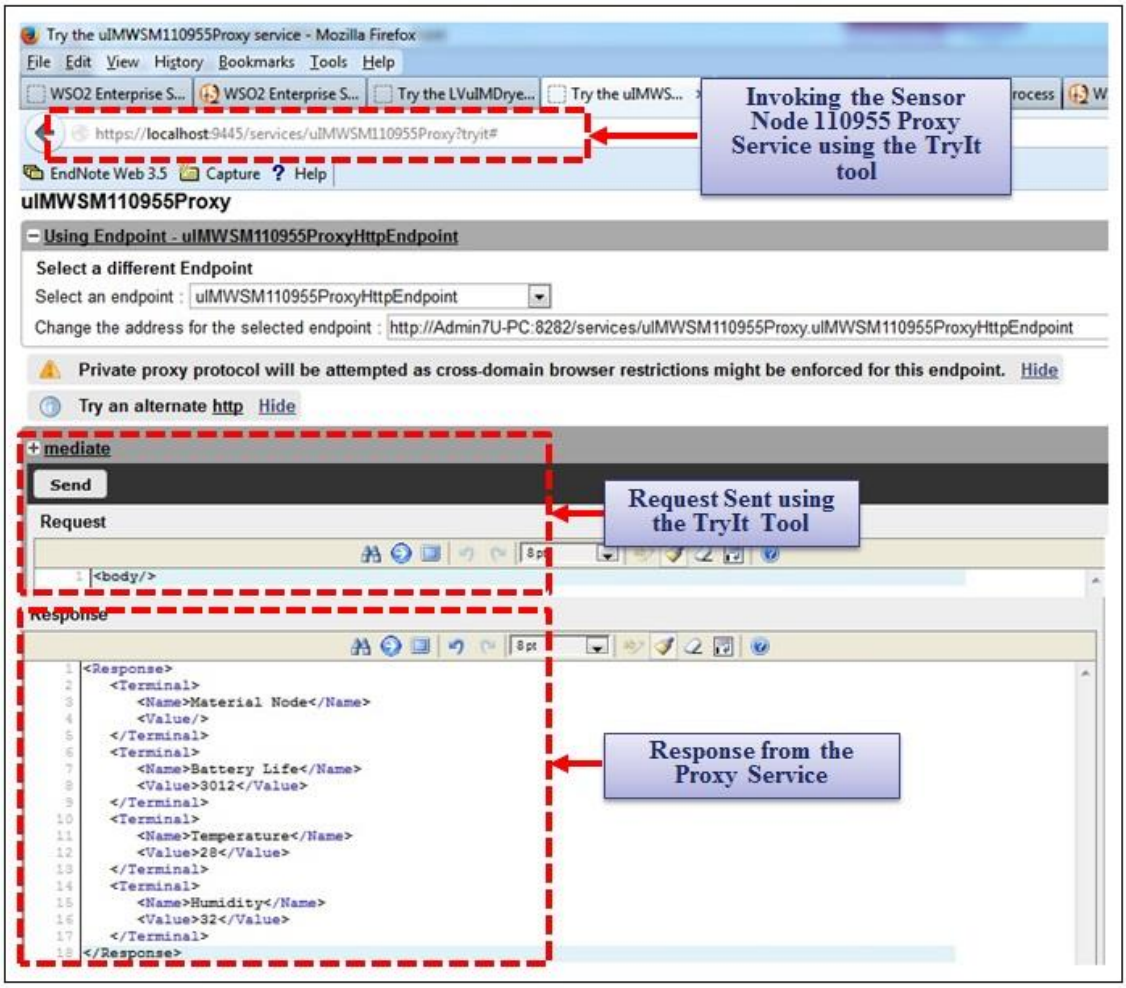


Figure E.1 WSO2 TryIt tool used to invoke a Web Service.

When the TryIt tool is selected direct from the tools sidebar a new window pops up asking the user to enter the URL of the WSDL file. Once a URL has been entered and the TryIt button has been clicked a new window appears with the available operations in the given WSDL file (Figure E.1). For the operations that take arguments, fields are available which allows primitive argument types. The values specified in these fields get passed on to the operations. If the operation

does not take any arguments a button appears that has the same name as the operation.

The other way of using the TryIt tool is when a service is deployed successfully on any WSO2 Carbon based server, a “Try this Service” link appears next to the service which is directly linked to the WSO2 TryIt tool. When this link is clicked the user is directed to the same window as shown in Figure E.1 where the operations available for the particular service are displayed. As before when the user gives the parameters for the operation, a button corresponding to the service appears as shown in Figure E.1. In this case for the Proxy Service the "mediate>>" button appears. When this button is clicked the proxy service gets invoked with the parameters specified in the primitive argument fields. The response from the invoked proxy service is displayed in the bottom window as shown in Figure E.1.

Appendix F. Business Processes Artefacts

F.1 Sensor Node WSDL file

The code below shows a single sensor node WSDL file. The WSDL file for other sensor nodes is very similar except for the sensor node address which is different and the services it offers.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions targetNamespace="http://polymer.bradford.org/LVWM/LVWM_11091a"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tns="http://polymer.bradford.org/LVWM/LVWM_11091a"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:ns1="http://polymer.bradford.org/LVWM/LVWM_11091a"
xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:types>
          <schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="unqualified"
                    targetNamespace="http://polymer.bradford.org/LVWM/LVWM_11091a">

         <element name="LVWM_11091aRequest">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>

         <element name="LVWM_11091aResponse">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>
      </schema>
   </wsdl:types>
   <wsdl:message name="LVWM_11091aIN"><wsdl:part element="ns1:LVWM_11091aRequest"
name="payload"/></wsdl:message>
   <wsdl:message name="LVWM_11091aOUT">
          <wsdl:part name="payload" type="xsd:string"></wsdl:part></wsdl:message>
   <wsdl:portType name="LVWM_11091aPortType">
      <wsdl:operation name="mediate">
         <wsdl:input message="tns:LVWM_11091aIN" wsaw:Action="urn:mediate" />
         <wsdl:output message="tns:LVWM_11091aOUT" wsaw:Action="urn:mediateResponse" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="LVWM_11091aSoap11Binding"
          type="tns:LVWM_11091aPortType">
          <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                    style="document" />
          <wsdl:operation name="mediate">
                    <soap:operation soapAction="urn:mediate" style="document" />
                    <wsdl:input>
                              <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                              <soap:body use="literal" />
```

```
                </wsdl:output>
            </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="LVWM_11091aSoap12Binding"
            type="tns:LVWM_11091aPortType">
            <wsdl:operation name="mediate">
                <soap12:operation soapAction="urn:mediate" style="document" />
                <wsdl:input>
                    <soap12:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal" />
                </wsdl:output>
            </wsdl:operation>
    </wsdl:binding>
    <!--    <wsdl:binding name="LVRProxyHttpBinding" type="tns:LVRProxyPortType">
        <http:binding verb="POST" />
        <wsdl:operation name="mediate">
            <http:operation location="mediate" />
            <wsdl:input>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding> -->

    <wsdl:binding name="LVWM_11091aHttpBinding"
            type="tns:LVWM_11091aPortType">
            <http:binding verb="GET" />
            <wsdl:operation name="mediate">
                <http:operation
                    location="/LabWS/11091a" />
                <wsdl:input>
                    <http:urlEncoded />
                </wsdl:input>
                <wsdl:output>
                    <mime:content type="text/xml" part="payload" />
                </wsdl:output>
            </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="LVWM_11091a">
        <wsdl:port name="LVWM_11091aHttpSoap11Endpoint"
binding="tns:LVWM_11091aSoap11Binding">
            <soap:address
location="http://localhost:8282/services/LVWM_11091a.LVWM_11091aHttpSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVWM_11091aHttpsSoap11Endpoint"
binding="tns:LVWM_11091aSoap11Binding">
            <soap:address
location="https://localhost:8245/services/LVWM_11091a.LVWM_11091aHttpsSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVWM_11091aHttpSoap12Endpoint"
binding="tns:LVWM_11091aSoap12Binding">
            <soap12:address
location="http://localhost:8282/services/LVWM_11091a.LVWM_11091aHttpSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVWM_11091aHttpsSoap12Endpoint"
binding="tns:LVWM_11091aSoap12Binding">
            <soap12:address
location="https://localhost:8245/services/LVWM_11091a.LVWM_11091aHttpsSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVWM_11091aHttpsEndpoint" binding="tns:LVWM_11091aHttpBinding">
            <http:address location="https://localhost:8085/LabWS/11091a"/>
        </wsdl:port>
```

```
      <wsdl:port name="LVWM_11091aHttpEndpoint" binding="tns:LVWM_11091aHttpBinding">
         <!--<http:address location="http://www.goodreads.com/author/list/4776766.xml"/>-->
            <http:address location="http://localhost:8085/LabWS/11091a"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

F.2 WSDL file for the LabVIEW Random Proxy Service

The LabVIEW based random proxy service running on the ESB server is invoked using this WSDL file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions targetNamespace="http://ws.apache.org/axis2"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:tns="http://ws.apache.org/axis2"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:ns1="http://ws.apache.org/axis2"
xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:types>
         <schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="unqualified"
                  targetNamespace="http://ws.apache.org/axis2">

         <element name="LVRandomRequest">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>

         <element name="LVRandomResponse">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>
      </schema>
   </wsdl:types>
   <wsdl:message name="in"><wsdl:part element="ns1:LVRandomRequest"
name="payload"/></wsdl:message>
   <wsdl:message name="out">
         <wsdl:part name="payload" type="xsd:string"></wsdl:part></wsdl:message>
   <wsdl:portType name="LVRandomProxyPortType">
      <wsdl:operation name="mediate">
         <wsdl:input message="ns1:in" wsaw:Action="urn:mediate" />
         <wsdl:output message="ns1:out" wsaw:Action="urn:mediateResponse" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="LVRandomProxySoap11Binding" type="tns:LVRandomProxyPortType">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
      <wsdl:operation name="mediate">
         <soap:operation soapAction="urn:mediate" style="document" />
         <wsdl:input>
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
```

```
    </wsdl:binding>
    <wsdl:binding name="LVRandomProxySoap12Binding" type="tns:LVRandomProxyPortType">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="mediate">
            <soap12:operation soapAction="urn:mediate" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
<!--    <wsdl:binding name="LVRProxyHttpBinding" type="tns:LVRProxyPortType">
        <http:binding verb="POST" />
        <wsdl:operation name="mediate">
            <http:operation location="mediate" />
            <wsdl:input>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding> -->

    <wsdl:binding name="LVRandomProxyHttpBinding" type="tns:LVRandomProxyPortType">
        <http:binding verb="GET"/>
        <wsdl:operation name="mediate">
            <http:operation location="/uIMPSimple/uIMPSimpleRandomWS/random"/>
            <wsdl:input>
                <http:urlEncoded/>
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="payload"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="LVRandomProxy">
        <wsdl:port name="LVRandomProxyHttpSoap11Endpoint"
binding="tns:LVRandomProxySoap11Binding">
            <soap:address
location="http://localhost:8282/services/LVRandomProxy.LVRandomProxyHttpSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVRandomProxyHttpsSoap11Endpoint"
binding="tns:LVRandomProxySoap11Binding">
            <soap:address
location="https://localhost:8245/services/LVRandomProxy.LVRandomProxyHttpsSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVRandomProxyHttpSoap12Endpoint"
binding="tns:LVRandomProxySoap12Binding">
            <soap12:address
location="http://localhost:8282/services/LVRandomProxy.LVRandomProxyHttpSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVRandomProxyHttpsSoap12Endpoint"
binding="tns:LVRandomProxySoap12Binding">
            <soap12:address
location="https://localhost:8245/services/LVRandomProxy.LVRandomProxyHttpsSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVRandomProxyHttpsEndpoint" binding="tns:LVRandomProxyHttpBinding">
            <http:address location="https://localhost:8085/uIMPSimple/uIMPSimpleRandomWS/random"/>
        </wsdl:port>
        <wsdl:port name="LVRandomProxyHttpEndpoint" binding="tns:LVRandomProxyHttpBinding">
            <!--<http:address location="http://www.goodreads.com/author/list/4776766.xml"/>-->
                <http:address location="http://localhost:8085/uIMPSimple/uIMPSimpleRandomWS/random"/>
        </wsdl:port>
```

```
    </wsdl:service>
</wsdl:definitions>
```

F.3 Deployment Descriptor File

A deployment descriptor file used for a the business process to be deployed on the WSO2 BPS server.

```
<?xml version="1.0" encoding="UTF-8"?>
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
xmlns:LVWM_110955="http://polymer.bradford.org/LVWM/LVWM_110955"
xmlns:LVWM_11ad58="http://polymer.bradford.org/LVWM/LVWM_11ad58"
xmlns:calculatorservices.polymer.brad.org="http://calculatorservices.polymer.brad.org"
xmlns:sample="http://polymer.bradford.org/bps/uIMPLVWSMProcess">
  <process name="sample:uIMPLVWSMProcess">
    <active>true</active>
    <retired>false</retired>
    <process-events generate="all"/>
    <provide partnerLink="client">
      <service name="sample:uIMPLVWSMProcess" port="uIMPLVWSMProcessPort"/>
    </provide>
    <invoke partnerLink="uIMP110955PL">
      <service name="LVWM_110955:LVWM_110955" port="LVWM_110955HttpEndpoint"/>
    </invoke>
    <invoke partnerLink="uIMPDryerPL">
      <service name="calculatorservices.polymer.brad.org:CalculatorService"
port="CalculatorServiceHttpSoap11Endpoint"/>
    </invoke>
    <invoke partnerLink="uIMP11ad58PL">
      <service name="LVWM_11ad58:LVWM_11ad58" port="LVWM_11ad58HttpEndpoint"/>
    </invoke>
  </process>
</deploy>
```

F.4 LVWSM Business Process Artefact XML File

The XML version of the business process artefact file created for the LabVIEW based dryer and temperature alarm simulations is given below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:plnk="http://docs.oasis-
open.org/wsbpel/2.0/plnktype" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://polymer.bradford.org/bps/uIMPLVWSMProcess" xmlns:vprop="http://docs.oasis-
open.org/wsbpel/2.0/varprop" xmlns:wsdl="http://polymer.bradford.org/LVWM/LVWM_110955"
xmlns:wsdl1="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:wsdl2="http://polymer.bradford.org/LVSIM/LVDryer" name="uIMPLVWSMProcess"
targetNamespace="http://polymer.bradford.org/bps/uIMPLVWSMProcess">

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of types participating in this BPEL process
    The BPEL Designer will generate default request and response types
    but you can define or import any XML Schema type and use them as part
```

```
    of the message types.
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
   <plnk:partnerLinkType name="uIMP110955PLT">
   <plnk:role name="uIMP110955Role" portType="wsdl:LVWM_110955PortType"/>
  </plnk:partnerLinkType>
   <plnk:partnerLinkType name="uIMPTAlarmPLT">
   <plnk:role name="uIMPTAlarmRole" portType="wsdl1:LVuIMTAlarmProxyPortType"/>
  </plnk:partnerLinkType>
   <plnk:partnerLinkType name="uIMPDryerPLT">
   <plnk:role name="uIMPDryerRole" portType="wsdl2:LVuIMPDryerProxyPortType"/>
  </plnk:partnerLinkType>
   <import location="LVWM_110955.wsdl"
namespace="http://polymer.bradford.org/LVWM/LVWM_110955"/>
   <import location="LVuIMDryerProxy.wsdl" namespace="http://polymer.bradford.org/LVSIM/LVDryer"/>
   <import location="LVuIMTAlarmProxy.wsdl"
namespace="http://polymer.bradford.org/LVSIM/LVTAlarm"/>
   <types>
      <schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://polymer.bradford.org/bps/uIMPLVWSMProcess">

         <element name="uIMPLVWSMProcessRequest">
            <complexType>
               <sequence>
                  <element name="input" type="string" minOccurs="0"/>
               </sequence>
            </complexType>
         </element>

         <element name="uIMPLVWSMProcessResponse">
            <complexType>
               <sequence>
                     <element name="DryerResult" type="string"
                              minOccurs="0" />
                     <element name="AlarmResult" type="string" minOccurs="0"></element>
               </sequence>
            </complexType>
         </element>
      </schema>
   </types>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
   <message name="uIMPLVWSMProcessRequestMessage">
      <part element="tns:uIMPLVWSMProcessRequest" name="payload"/>
   </message>
   <message name="uIMPLVWSMProcessResponseMessage">
      <part element="tns:uIMPLVWSMProcessResponse" name="payload"/>
   </message>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->

   <!-- portType implemented by the uIMPLVWSMProcess BPEL process -->
   <portType name="uIMPLVWSMProcess">
      <operation name="process">
         <input message="tns:uIMPLVWSMProcessRequestMessage"/>
         <output message="tns:uIMPLVWSMProcessResponseMessage"/>
      </operation>
   </portType>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION
```

```
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
   <plnk:partnerLinkType name="uIMPLVWSMProcess">
      <plnk:role name="uIMPLVWSMProcessProvider" portType="tns:uIMPLVWSMProcess"/>
   </plnk:partnerLinkType>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    BINDING DEFINITION - Defines the message format and protocol details
    for a web service.
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
   <binding name="uIMPLVWSMProcessBinding" type="tns:uIMPLVWSMProcess">
          <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="process">
                 <soap:operation
soapAction="http://polymer.bradford.org/bps/uIMPLVWSMProcess/process"/>
                 <input>
                        <soap:body use="literal"/>
                 </input>
                 <output>
                        <soap:body use="literal"/>
                 </output>
          </operation>
   </binding>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    SERVICE DEFINITION - A service groups a set of ports into
    a service unit.
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
   <service name="uIMPLVWSMProcess">
          <port binding="tns:uIMPLVWSMProcessBinding" name="uIMPLVWSMProcessPort">
                 <soap:address location="http://localhost:9766/uIMPLVWSMProcess"/>
          </port>
   </service>
</definitions>
```

F.4.1 LVWSM Business Process XML File

The complete XML version of the business process file created for the LabVIEW based dryer and temperature alarm simulations is given below.

```
<!-- uIMPLVWSMProcess BPEL Process [Generated by the Eclipse BPEL Designer] -->
<!-- Date: Mon Mar 05 12:13:11 IST 2012 -->
<bpel:process name="uIMPLVWSMProcess"
       targetNamespace="http://polymer.bradford.org/bps/uIMPLVWSMProcess"
       suppressJoinFailure="yes"
       xmlns:tns="http://polymer.bradford.org/bps/uIMPLVWSMProcess"
       xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
       xmlns:ns="http://polymer.bradford.org/LVWM/LVWM_110955"
xmlns:ns0="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:ns1="http://polymer.bradford.org/LVSIM/LVDryer">

   <!-- Import the client WSDL -->
   <bpel:import namespace="http://polymer.bradford.org/LVWM/LVWM_110955"
location="LVWM_110955.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://polymer.bradford.org/LVSIM/LVDryer"
location="LVuIMDryerProxy.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://polymer.bradford.org/LVSIM/LVTAlarm"
location="LVuIMTAlarmProxy.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import location="uIMPLVWSMProcessArtifacts.wsdl"
namespace="http://polymer.bradford.org/bps/uIMPLVWSMProcess"
                importType="http://schemas.xmlsoap.org/wsdl/" />

   <!-- ================================================================= -->
   <!-- PARTNERLINKS                                                      -->
```

```
   <!-- List of services participating in this BPEL process            -->
   <!-- ================================================================= -->
   <bpel:partnerLinks>
      <!-- The 'client' role represents the requester of this service. -->
      <bpel:partnerLink name="client"
                  partnerLinkType="tns:uIMPLVWSMProcess"
                  myRole="uIMPLVWSMProcessProvider"
                  />

      <bpel:partnerLink name="uIMP110955PL" partnerLinkType="tns:uIMP110955PLT"
partnerRole="uIMP110955Role"></bpel:partnerLink>

      <bpel:partnerLink name="uIMPTAlarmPL" partnerLinkType="tns:uIMPTAlarmPLT"
partnerRole="uIMPTAlarmRole"></bpel:partnerLink>
      <bpel:partnerLink name="uIMPDryerPL" partnerLinkType="tns:uIMPDryerPLT"
partnerRole="uIMPDryerRole"></bpel:partnerLink>
   </bpel:partnerLinks>

   <!-- ================================================================= -->
   <!-- VARIABLES                                                         -->
   <!-- List of messages and XML documents used within this BPEL process  -->
   <!-- ================================================================= -->
   <bpel:variables>
      <!-- Reference to the message passed as input during initiation -->
      <bpel:variable name="LVWMProcessIn"
            messageType="tns:uIMPLVWSMProcessRequestMessage"/>

      <!--
        Reference to the message that will be returned to the requester
        -->
      <bpel:variable name="LVWMProcessOut"
messageType="tns:uIMPLVWSMProcessResponseMessage"/>
      <bpel:variable name="uIMP110955Out" messageType="ns:LVWM_110955OUT"></bpel:variable>
      <bpel:variable name="uIMP110955In" messageType="ns:LVWM_110955IN"></bpel:variable>
      <bpel:variable name="uIMPTAlarmOut" messageType="ns0:TAlarmOUT"></bpel:variable>
      <bpel:variable name="uIMPTAlarmIn" messageType="ns0:TAlarmIN"></bpel:variable>
      <bpel:variable name="uIMPDryerOut" messageType="ns1:DryerOUT"></bpel:variable>
      <bpel:variable name="uIMPDryerIn" messageType="ns1:DryerIN"></bpel:variable>

      <bpel:variable name="uIMPTAlarmPLResponse"
messageType="ns0:mediateResponse"></bpel:variable>
      <bpel:variable name="uIMPTAlarmPLRequest"
messageType="ns0:mediateRequest"></bpel:variable>
      <bpel:variable name="uIMPDryerPLResponse"
messageType="ns1:mediateResponse"></bpel:variable>
      <bpel:variable name="uIMPDryerPLRequest" messageType="ns1:mediateRequest"></bpel:variable>
      <bpel:variable name="LVWMAlarmOut"
messageType="tns:uIMPLVWSMProcessResponseMessage"></bpel:variable>
   </bpel:variables>

   <!-- ================================================================= -->
   <!-- ORCHESTRATION LOGIC                                               -->
   <!-- Set of activities coordinating the flow of messages across the    -->
   <!-- services integrated within this business process                  -->
   <!-- ================================================================= -->
   <bpel:sequence name="main">

      <!-- Receive input from requester.
          Note: This maps to operation defined in uIMPLVWSMProcess.wsdl
          -->
      <bpel:receive name="ProcessInput" partnerLink="client"
            portType="tns:uIMPLVWSMProcess"
            operation="process" variable="LVWMProcessIn"
            createInstance="yes"/>

      <!-- Generate reply to synchronous request -->
      <bpel:assign validate="no" name="Assignto_mWS">
```

```
            <bpel:copy>
               <bpel:from>
                  <bpel:literal>
                     <ns1:LVWM_110955Request
xmlns:ns1="http://polymer.bradford.org/LVWM/LVWM_110955"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                     </ns1:LVWM_110955Request>
                  </bpel:literal>
               </bpel:from>
               <bpel:to variable="uIMP110955In" part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
               <bpel:from part="payload" variable="LVWMProcessIn">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:input]]></bpel:query>
               </bpel:from>
               <bpel:to part="payload" variable="uIMP110955In"></bpel:to>
            </bpel:copy>
         </bpel:assign>

         <bpel:invoke name="InvokeLVWM110955" partnerLink="uIMP110955PL" operation="mediate"
         portType="ns:LVWM_110955PortType" inputVariable="uIMP110955In"
outputVariable="uIMP110955Out"></bpel:invoke>

         <bpel:if name="If">

<bpel:condition><![CDATA[bpel:getVariableData('uIMP110955Out','payload')/Response/Terminal[3]/Value
> 27]]></bpel:condition>
            <bpel:sequence name="TempGT27C">
               <bpel:assign validate="no" name="AssignTo_AlarmService">
                  <bpel:copy>
                     <bpel:from>
                        <bpel:literal>
                           <ns1:LVTAlarmRequest xmlns:ns1="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                           </ns1:LVTAlarmRequest>
                        </bpel:literal>
                     </bpel:from>
                     <bpel:to variable="uIMPTAlarmIn" part="payload"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="payload" variable="uIMP110955Out"></bpel:from>
                     <bpel:to part="payload" variable="uIMPTAlarmIn"></bpel:to>
                  </bpel:copy>
               </bpel:assign>

         <bpel:invoke name="InvokeTempAlarmService" partnerLink="uIMPTAlarmPL" operation="mediate"
portType="ns0:LVuIMTAlarmProxyPortType" inputVariable="uIMPTAlarmPLRequest"
outputVariable="uIMPTAlarmPLResponse"></bpel:invoke>
         <bpel:assign validate="no" name="AssignTo_AlarmOut">
                  <bpel:copy>
                     <bpel:from>
                        <bpel:literal>
                           <tns:uIMPLVWSMProcessResponse
xmlns:tns="http://polymer.bradford.org/bps/uIMPLVWSMProcess"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><tns:DryerResult>tns:DryerResult</tns:DryerResult>
  <tns:AlarmResult>tns:AlarmResult</tns:AlarmResult>

                           </tns:uIMPLVWSMProcessResponse>
                        </bpel:literal>
                     </bpel:from>
                     <bpel:to variable="LVWMAlarmOut" part="payload"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="payload" variable="uIMPTAlarmOut"></bpel:from>
```

```
                    <bpel:to part="payload" variable="LVWMAlarmOut">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:AlarmResult]]></bpel:q
uery>
                    </bpel:to>
                </bpel:copy>
            </bpel:assign>
            <bpel:reply name="AlarmOut" partnerLink="client" operation="process"
portType="tns:uIMPLVWSMProcess" variable="LVWMAlarmOut"></bpel:reply>
        </bpel:sequence>
    <bpel:elseif>

<bpel:condition><![CDATA[bpel:getVariableData('uIMP110955Out','payload')/Response/Terminal[4]/Value
> 30]]></bpel:condition>
            <bpel:sequence name="HumidityGT30">
                <bpel:assign validate="no" name="AssignTo_DryerService">
                    <bpel:copy>
                        <bpel:from>
                            <bpel:literal>
                                <ns1:LVDryerRequest xmlns:ns1="http://polymer.bradford.org/LVSIM/LVDryer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                                </ns1:LVDryerRequest>
                            </bpel:literal>
                        </bpel:from>
                        <bpel:to variable="uIMPDryerIn" part="payload"></bpel:to>
                    </bpel:copy>
                    <bpel:copy>
                        <bpel:from part="payload" variable="uIMP110955Out"></bpel:from>
                        <bpel:to part="payload" variable="uIMPDryerIn"></bpel:to>
                    </bpel:copy>
                </bpel:assign>
                <bpel:invoke name="InvokeLVDryerService" partnerLink="uIMPDryerPL" operation="mediate"
portType="ns1:LVuIMPDryerProxyPortType" inputVariable="uIMPDryerPLRequest"
outputVariable="uIMPDryerPLResponse"></bpel:invoke>

                <bpel:assign validate="no" name="AssignTo_DryerOut">
                    <bpel:copy>
                        <bpel:from>
                            <bpel:literal>
                                <tns:uIMPLVWSMProcessResponse
xmlns:tns="http://polymer.bradford.org/bps/uIMPLVWSMProcess"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><tns:DryerResult>tns:DryerResult</tns:DryerResult>
  <tns:AlarmResult>tns:AlarmResult</tns:AlarmResult>

                                </tns:uIMPLVWSMProcessResponse>
                            </bpel:literal>
                        </bpel:from>
                        <bpel:to variable="LVWMProcessOut" part="payload"></bpel:to>
                    </bpel:copy>
                    <bpel:copy>
                        <bpel:from part="payload" variable="uIMPDryerOut"></bpel:from>
                        <bpel:to part="payload" variable="LVWMProcessOut">
                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:DryerResult]]></bpel:qu
ery>
                        </bpel:to>
                    </bpel:copy>
                </bpel:assign>
                <bpel:reply name="DryerOut" partnerLink="client" operation="process"
portType="tns:uIMPLVWSMProcess" variable="LVWMProcessOut"></bpel:reply>
            </bpel:sequence>
        </bpel:elseif></bpel:if>

    </bpel:sequence>
</bpel:process>
```

F.4.2 LVDryerProxy WSDL File

The WSDL file used to invoke the LabVIEW Dryer proxy service running on the ESB server is shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions targetNamespace="http://polymer.bradford.org/LVSIM/LVDryer"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tns="http://polymer.bradford.org/LVSIM/LVDryer"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:ns1="http://polymer.bradford.org/LVSIM/LVDryer"
xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:types>
          <schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="unqualified"
                    targetNamespace="http://polymer.bradford.org/LVSIM/LVDryer">

         <element name="LVDryerRequest">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>

         <element name="LVDryerResponse">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>
         <element name="mediateResponse">
           <complexType>
                    <sequence>
                              <element name="out" type="string"></element>
                    </sequence>
           </complexType>
         </element>
          </schema>
   </wsdl:types>
   <wsdl:message name="DryerIN"><wsdl:part element="ns1:LVDryerRequest"
name="payload"/></wsdl:message>
   <wsdl:message name="DryerOUT">
          <wsdl:part name="payload" type="xsd:string"></wsdl:part></wsdl:message>
    <wsdl:message name="mediateRequest">
          <wsdl:part name="payload" element="ns1:LVDryerRequest"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="mediateResponse">
      <wsdl:part name="payload" type="xsd:string"></wsdl:part>
   </wsdl:message>
   <wsdl:portType name="LVuIMPDryerProxyPortType">
          <wsdl:operation name="mediate">
                    <wsdl:input message="tns:mediateRequest"></wsdl:input>
                    <wsdl:output message="tns:mediateResponse"></wsdl:output>
          </wsdl:operation>
   </wsdl:portType>
```

```
    <wsdl:binding name="LVuIMPDryerProxySoap11Binding" type="ns1:LVuIMPDryerProxyPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="mediate">
            <soap:operation soapAction="urn:mediate" style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="LVuIMPDryerProxySoap12Binding" type="ns1:LVuIMPDryerProxyPortType">
            <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
                        style="document" />
    </wsdl:binding>
    <!--    <wsdl:binding name="LVRProxyHttpBinding" type="tns:LVRProxyPortType">
        <http:binding verb="POST" />
        <wsdl:operation name="mediate">
            <http:operation location="mediate" />
            <wsdl:input>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding> -->

    <wsdl:binding name="LVuIMPDryerProxyHttpBinding"
            type="tns:LVuIMPDryerProxyPortType">
            <http:binding verb="GET" />
            <wsdl:operation name="mediate">
                        <http:operation location="/mediate" />
                        <wsdl:input>
                                    <http:urlEncoded />
                        </wsdl:input>
                        <wsdl:output>
                                    <mime:content type="text/xml" />
                        </wsdl:output>
            </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="LVuIMPDryerProxy">
        <wsdl:port name="LVuIMPDryerProxyHttpSoap11Endpoint"
binding="tns:LVuIMPDryerProxySoap11Binding">
            <soap:address
location="http://localhost:8282/services/LVuIMPDryerProxy.LVuIMPDryerProxyHttpSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMPDryerProxyHttpsSoap11Endpoint"
binding="tns:LVuIMPDryerProxySoap11Binding">
            <soap:address location="https://localhost:8245/services/LVuIMPDryerHttpsSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMPDryerProxyHttpSoap12Endpoint"
binding="tns:LVuIMPDryerProxySoap12Binding">
            <soap12:address
location="http://localhost:8282/services/LVuIMPDryerProxy.LVuIMPDryerProxyHttpSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMPDryerProxyHttpsSoap12Endpoint"
binding="tns:LVuIMPDryerProxySoap12Binding">
            <soap12:address
location="https://localhost:8245/services/LVuIMPDryerProxy.LVuIMPDryerProxyHttpsSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMPDryerProxyHttpsEndpoint" binding="ns1:LVuIMPDryerProxyHttpBinding">
            <http:address location="https://localhost:8085/uIMPBPEL/uIMPBPELDryerWS/Dryer"/>
        </wsdl:port>
        <wsdl:port name="LVuIMPDryerProxyHttpEndpoint" binding="ns1:LVuIMPDryerProxyHttpBinding">
```

```
        <!--<http:address location="http://www.goodreads.com/author/list/4776766.xml"/>-->
            <http:address location="http://localhost:8085/uIMPBPEL/uIMPBPELDryerWS/Dryer"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

F.4.3 LVTAlarmProxy WSDL File

The WSDL file used to invoke the LabVIEW Temperature Alarm proxy service running on the ESB server is shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<wsdl:definitions targetNamespace="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tns="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:ns1="http://polymer.bradford.org/LVSIM/LVTAlarm"
xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:types>
         <schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="unqualified"
                  targetNamespace="http://polymer.bradford.org/LVSIM/LVTAlarm">

         <element name="LVTAlarmRequest">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>

         <element name="LVTAlarmResponse">
            <complexType>
               <sequence>
                  <any minOccurs="0"/>
               </sequence>
            </complexType>
         </element>
         <element name="NewOperation">
           <complexType>
                  <sequence>
                        <element name="in" type="string"></element>
                  </sequence>
           </complexType>
         </element>
         <element name="NewOperationResponse">
           <complexType>
                  <sequence>
                        <element name="out" type="string"></element>
                  </sequence>
           </complexType>
         </element>
           </schema>
   </wsdl:types>
   <wsdl:message name="TAlarmIN"><wsdl:part element="ns1:LVTAlarmRequest"
name="payload"/></wsdl:message>
   <wsdl:message name="TAlarmOUT">
         <wsdl:part name="payload" type="xsd:string"></wsdl:part></wsdl:message>
   <wsdl:message name="mediateRequest">
```

```
            <wsdl:part name="payload" element="ns1:LVTAlarmRequest"></wsdl:part>
    </wsdl:message>
    <wsdl:message name="mediateResponse">
        <wsdl:part name="payload" type="xsd:string"></wsdl:part>
    </wsdl:message>
    <wsdl:portType name="LVuIMTAlarmProxyPortType">
            <wsdl:operation name="mediate">
                    <wsdl:input message="tns:mediateRequest"></wsdl:input>
                    <wsdl:output message="tns:mediateResponse"></wsdl:output>
            </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="LVuIMTAlarmProxySoap11Binding" type="tns:LVuIMTAlarmProxyPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="mediate">
            <soap:operation soapAction="urn:mediate" style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="LVuIMTAlarmProxySoap12Binding" type="ns1:LVuIMTAlarmProxyPortType">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="mediate">
            <soap12:operation soapAction="urn:mediate" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
  <!--    <wsdl:binding name="LVuIMTAlarmProxyHttpBinding" type="tns:LVuIMTAlarmProxyPortType">
        <http:binding verb="POST" />
        <wsdl:operation name="mediate">
            <http:operation location="mediate" />
            <wsdl:input>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="parameters" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding> -->

    <wsdl:binding name="LVuIMTAlarmProxyHttpBinding"
            type="tns:LVuIMTAlarmProxyPortType">
            <http:binding verb="GET" />
            <wsdl:operation name="mediate">
                    <http:operation location="/uIMPBPEL/uIMPTAlarmWS/TAlarm" />
                    <wsdl:input>
                            <http:urlEncoded />
                    </wsdl:input>
                    <wsdl:output>
                            <mime:content type="text/xml" part="payload" />
                    </wsdl:output>
            </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="LVuIMTAlarmProxy">
        <wsdl:port name="LVuIMTAlarmProxyHttpSoap11Endpoint"
binding="ns1:LVuIMTAlarmProxySoap11Binding">
            <soap:address
location="http://localhost:8282/services/LVuIMTAlarmProxy.LVuIMTAlarmHttpSoap11Endpoint"/>
```

```
        </wsdl:port>
        <wsdl:port name="LVuIMTAlarmProxyHttpsSoap11Endpoint"
binding="ns1:LVuIMTAlarmProxySoap11Binding">
            <soap:address
location="https://localhost:8245/services/LVuIMTAlarmProxy.LVuIMTAlarmProxyHttpsSoap11Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMTAlarmProxyHttpSoap12Endpoint"
binding="ns1:LVuIMTAlarmProxySoap12Binding">
            <soap12:address
location="http://localhost:8282/services/LVLVuIMPDryerProxy.LVLVuIMPDryerProxyHttpSoap12Endpoint"
/>
        </wsdl:port>
        <wsdl:port name="LVuIMTAlarmProxyHttpsSoap12Endpoint"
binding="ns1:LVuIMTAlarmProxySoap12Binding">
            <soap12:address
location="https://localhost:8245/services/LVuIMPDryerProxy.LVuIMPDryerProxyHttpsSoap12Endpoint"/>
        </wsdl:port>
        <wsdl:port name="LVuIMTAlarmProxyHttpsEndpoint" binding="ns1:LVuIMTAlarmProxyHttpBinding">
            <http:address location="https://localhost:8085/uIMPBPEL/uIMPBPELTAlarmWS/Alarm"/>
        </wsdl:port>
        <wsdl:port name="LVuIMTAlarmHttpEndpoint" binding="ns1:LVuIMTAlarmProxyHttpBinding">
            <!--<http:address location="http://www.goodreads.com/author/list/4776766.xml"/>-->
                <http:address location="http://localhost:8085/uIMPBPEL/uIMPBPELTAlarmWS/Alarm"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```